KEYCLOAK

(
Ref :
https://www.youtube.com/watch?v=6xaNsACIq0s&list=PLHXvj3cRjbzs8TaT-RX1qJYYK2MjRro-P
YouTube : Dive Into Development)
For Oauth Dependencies (to enable Oauth services)

```
<dependency>
                  <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
            </dependency>
```

For Keycloak Dependencies:
Keycloak Version and Dependencies version need to be same

```
<!--This will be Deprecated soon in keycloak and In spring boot

https://mvnrepository.com/artifact/org.keycloak/keycloak-spring-boot-starter
-->
            <!--<dependency>
                  <groupId>org.keycloak</groupId>
                  <artifactId>keycloak-spring-boot-starter</artifactId>
                  <version>23.0.4</version>
            </dependency>-->
            <!--

https://mvnrepository.com/artifact/org.keycloak/keycloak-servlet-filter-adapt
er -->
            <!--<dependency>
                  <groupId>org.keycloak</groupId>
                  <artifactId>keycloak-servlet-filter-adapter</artifactId>
                  <version>23.0.4</version>
                  <scope>provided</scope>
            </dependency>-->
            <!--

https://mvnrepository.com/artifact/org.keycloak/keycloak-policy-enforcer -->
            <dependency>
                  <groupId>org.keycloak</groupId>
                  <artifactId>keycloak-policy-enforcer</artifactId>
                  <version>23.0.4</version>
            </dependency>
```

```
		<!--
https://mvnrepository.com/artifact/org.keycloak/keycloak-core -->
		<dependency>
			<groupId>org.keycloak</groupId>
			<artifactId>keycloak-core</artifactId>
			<version>23.0.4</version>
		</dependency>
		<!--

https://mvnrepository.com/artifact/org.keycloak/keycloak-admin-client -->
		<dependency>
			<groupId>org.keycloak</groupId>
			<artifactId>keycloak-admin-client</artifactId>
			<version>23.0.4</version>
		</dependency>
```

To Store in your Own Database Go to Keycloak Folder -> /keycloak/conf -> open keycloak.conf change the DB Details it will store DB automatically.

Authentication & Authorization Flow in Various Scenarios

Resource request:

It is used to send the request and needs to be authenticated by the authentication server and authentication token needs to be sent to the Resource server.

Eg: UI App,Postman,Client Server,Angular,Swagger UI.

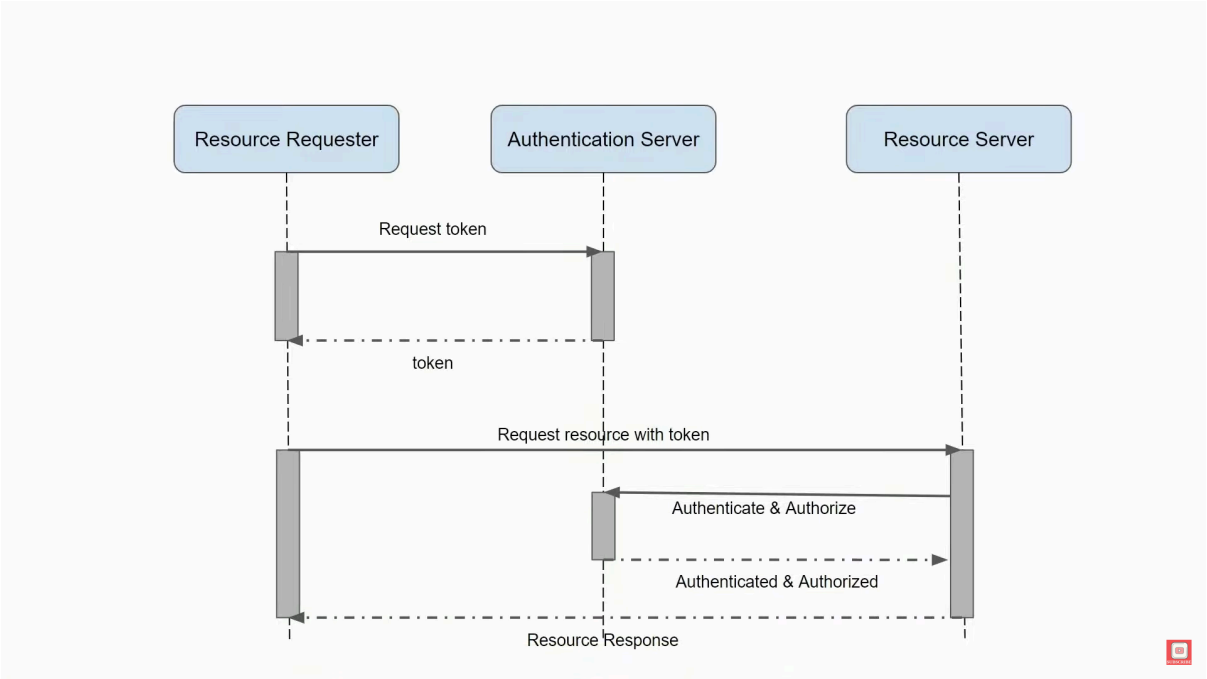
Authentication server:

It will check the authenticate the resource request API and send it with Token generation,Refresh Token,Exp Date etc…
and check if the token is valid or not for every request.

Eg: Keycloak,Okta,Asure and my own spring auth server.

Resource Server:

It will contain the data and send a resource request.
Eg: our spring boot and Quarkus etc..



KEYCLOAK CONFIG

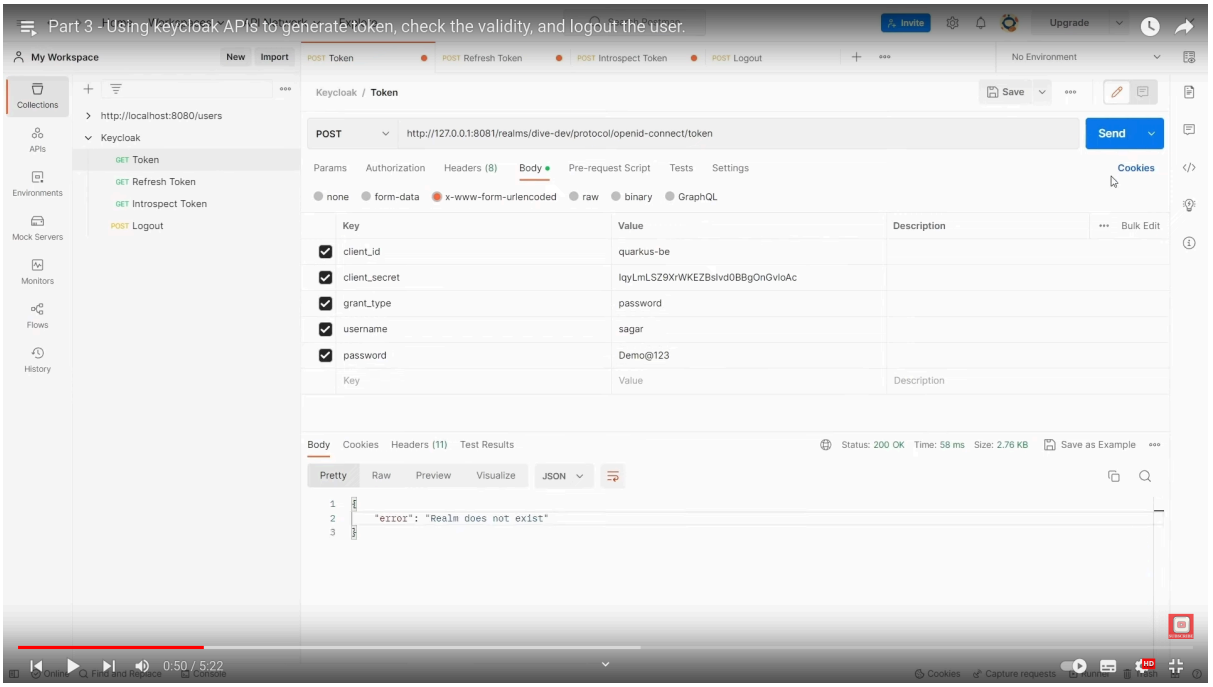
Create Client ,Enter the name of client,click the create.

Client authentication → It will enable the credential key.
Authorization →
Standard Flow → redirect the login temp and redirect URL tab will be open.
Direct Access Flow → Mostly used for REST Api Calls.
Implicit Flow →
Server Account Flow →
OAuth 2.0 Device Authorization Flow →
OIDC CIBA Grant →

Getting Token:

Token_endpoint : http://localhost:8081/realms/quarkus-be/protocol/openid-connect/token

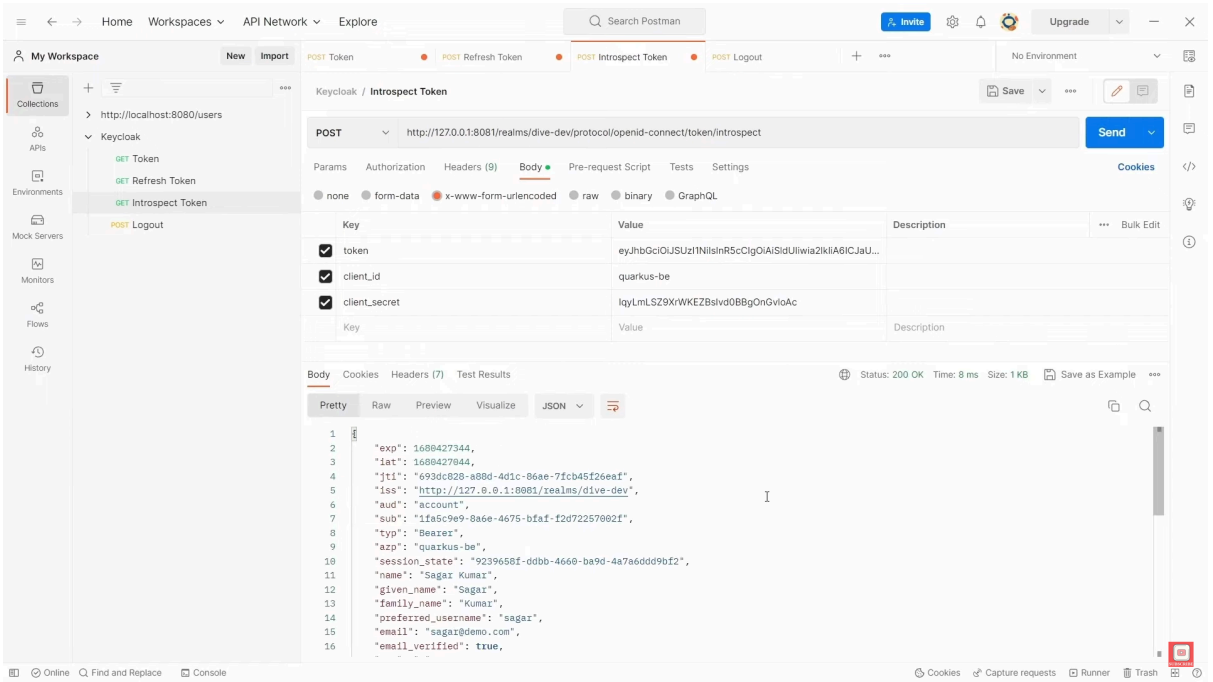
client_id
client_secret
grant_type
username
password



To Check Token is Active or Not :

Introspection_endpoint :
http://localhost:8081/realms/quarkus-be/protocol/openid-connect/token/introspect

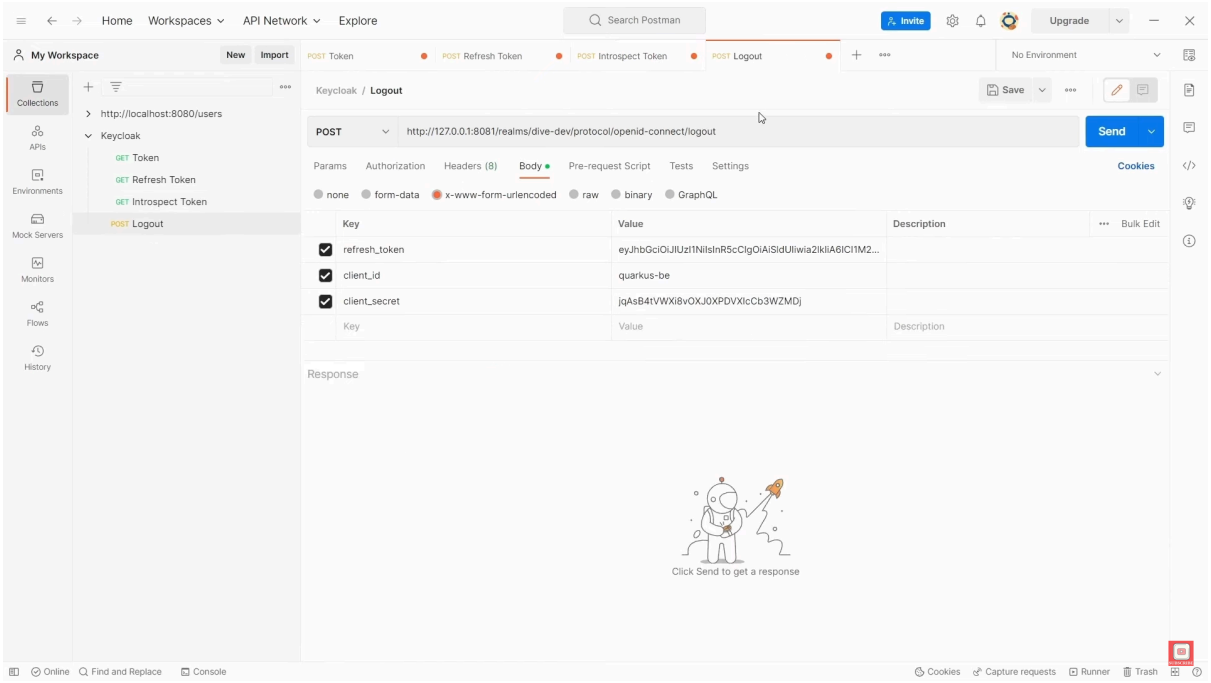
token
client_id
client_secret



To Logout:

Logout_endpoint : http://localhost:8081/realms/quarkus-be/protocol/openid-connect/logout

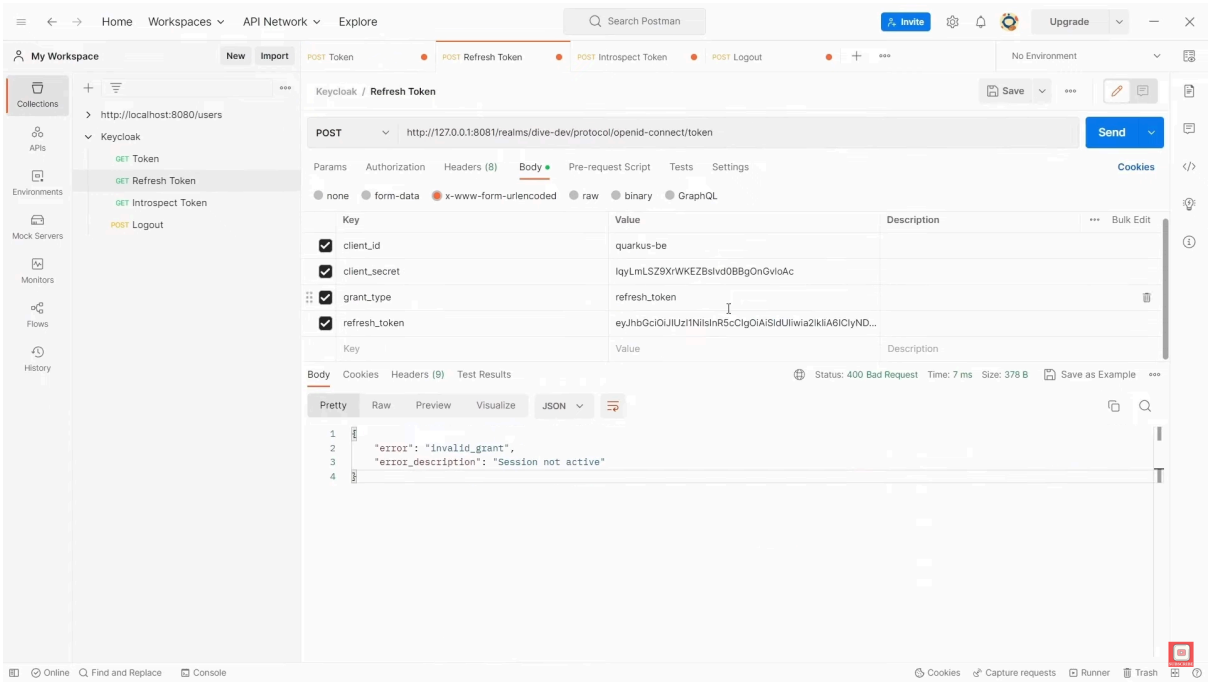
refresh_token
client_id
client_secret



Refresh Token to generate valid Token:

Token API : http://localhost:8081/realms/quarkus-be/protocol/openid-connect/token

client_id
client_secret
grant_type
refresh_token



We can used to add Extra details in token using client scope

Expanding token limit and refresh token limit :

For Refresh token : realm Setting -> session -> IN SSO Session Idle change it.
For Token : realm Setting -> Token -> In Access Token Lifespan change it.

If Token is Expired Refresh Token can be Generated only 3 or 4 Time (which means we can set Limits)

In realm Setting -> Token -> Revoke Refresh Token Enable it. It shows 0 which means it will be considered as 1.

SPRING BOOT AND KEYCLOAK

Authentication & Authorization In JWT & OPAQUE TOKEN

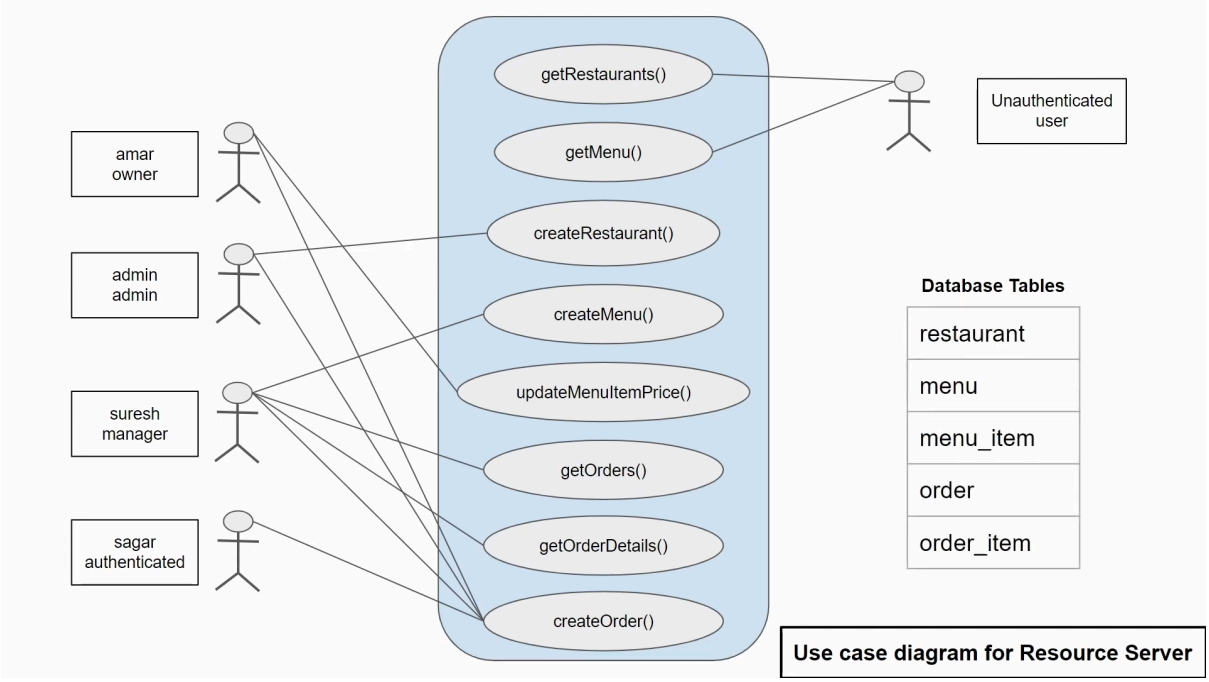
Opaque tokens and JWT (JSON Web Tokens) are both methods used in authentication and authorization systems, but they serve slightly different purposes and have different characteristics. Here's a brief overview of each:

1. **Opaque Tokens:**
    - **An opaque token is typically a randomly generated string that serves as a reference to some server-side data.**

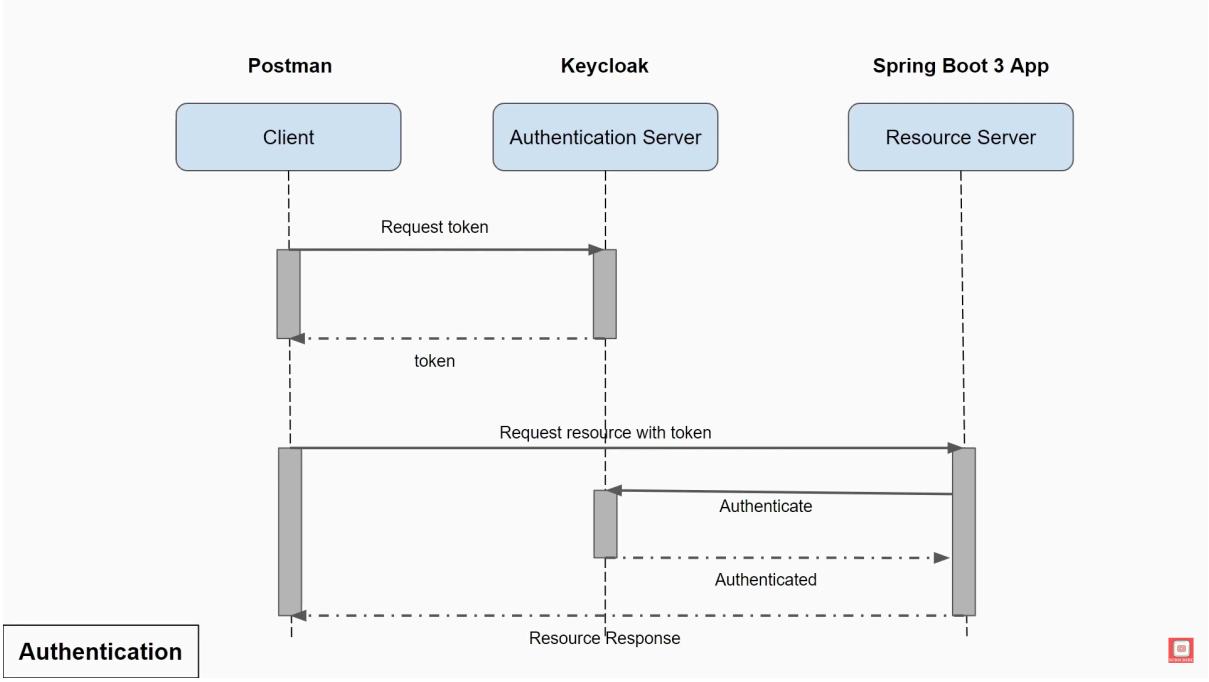
- **When a client sends an opaque token to a server, the server must validate the token by looking up its associated data in a database or some other storage mechanism.**
- **The client typically doesn't have any information about the contents of the token and cannot decode it or extract any meaningful information from it.**
- **Opaque tokens are generally used in scenarios where the server needs to maintain control over the data associated with the token and wants to minimize the amount of information sent over the network.**
- **They are commonly used in sessions-based authentication systems, where the server maintains session state on behalf of the client.**

2. **JWT Tokens:**
   - **JWT is a compact, URL-safe means of representing claims to be transferred between two parties.**
   - **JWTs are encoded as JSON objects and are self-contained, meaning they contain all the necessary information about the user or session within the token itself.**
   - **JWTs typically consist of three parts: a header, a payload (claims), and a signature. The payload contains the claims, such as user ID, expiration time, etc.**
   - **JWTs are often used in stateless authentication systems, where the server does not need to maintain session state. Since JWTs contain all the necessary information, the server can validate and process them without needing to look up additional data.**
   - **JWTs are commonly used in token-based authentication systems, where the client sends the token with each request to authenticate itself.**

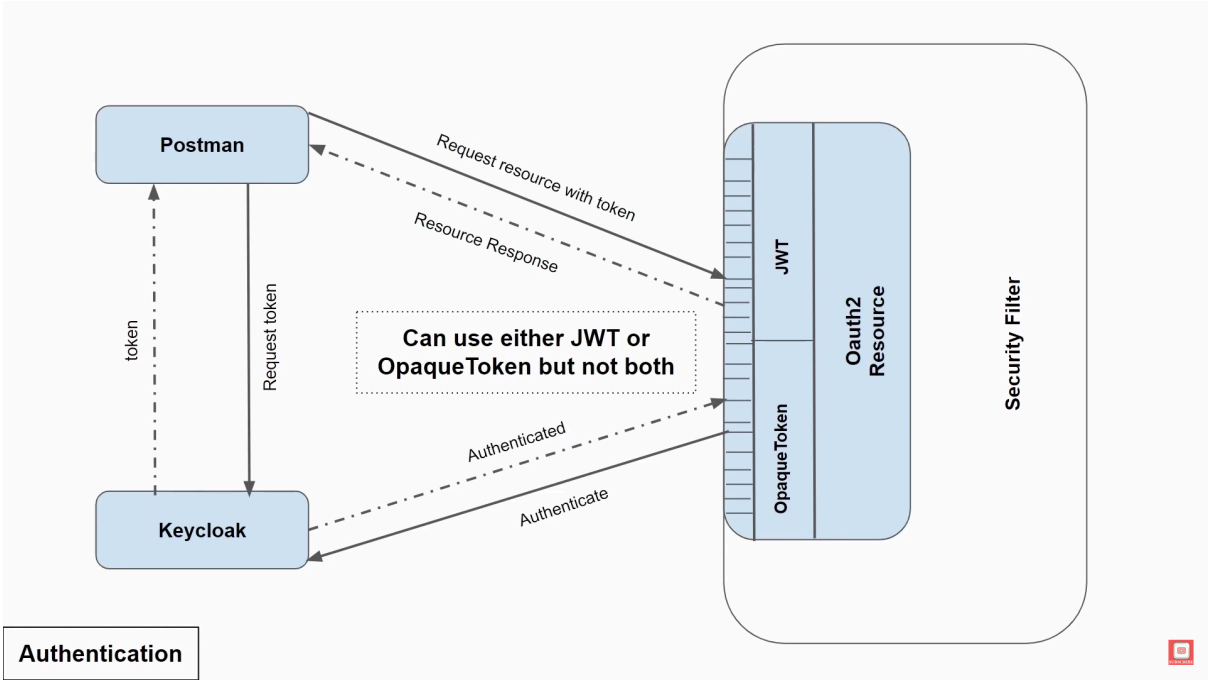
Create Spring boot Application:



The Flow of Client to Resource Server.



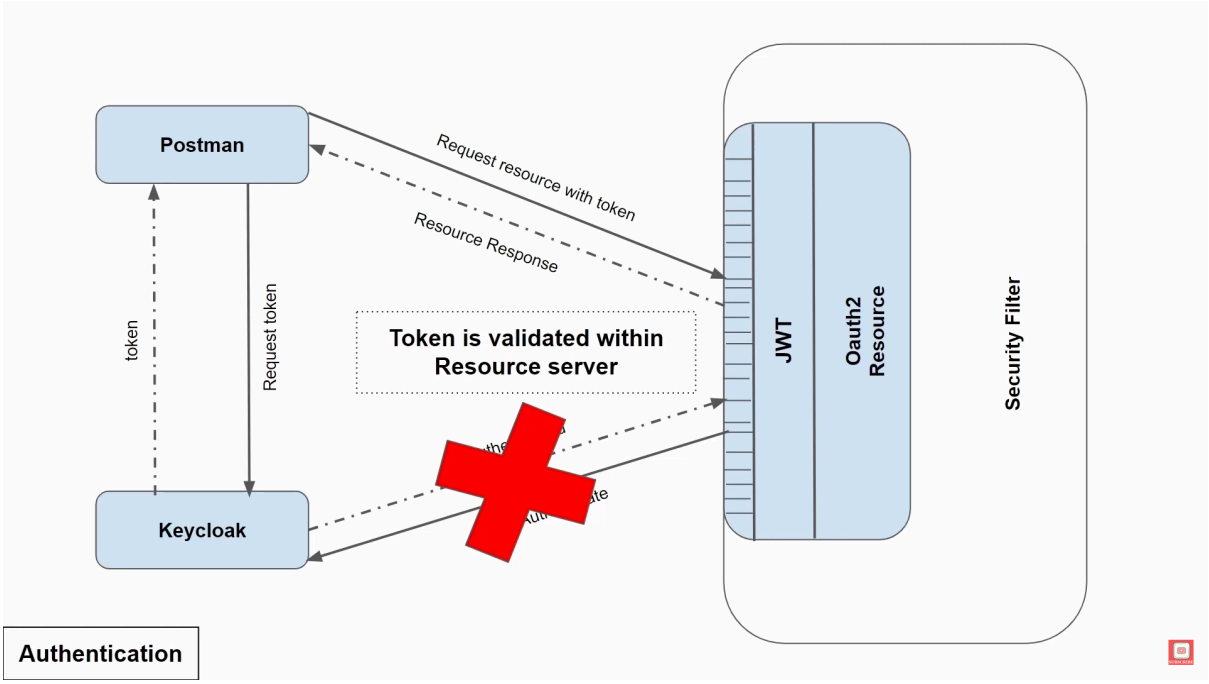
This is another way of Flow for Client to Resource Server.



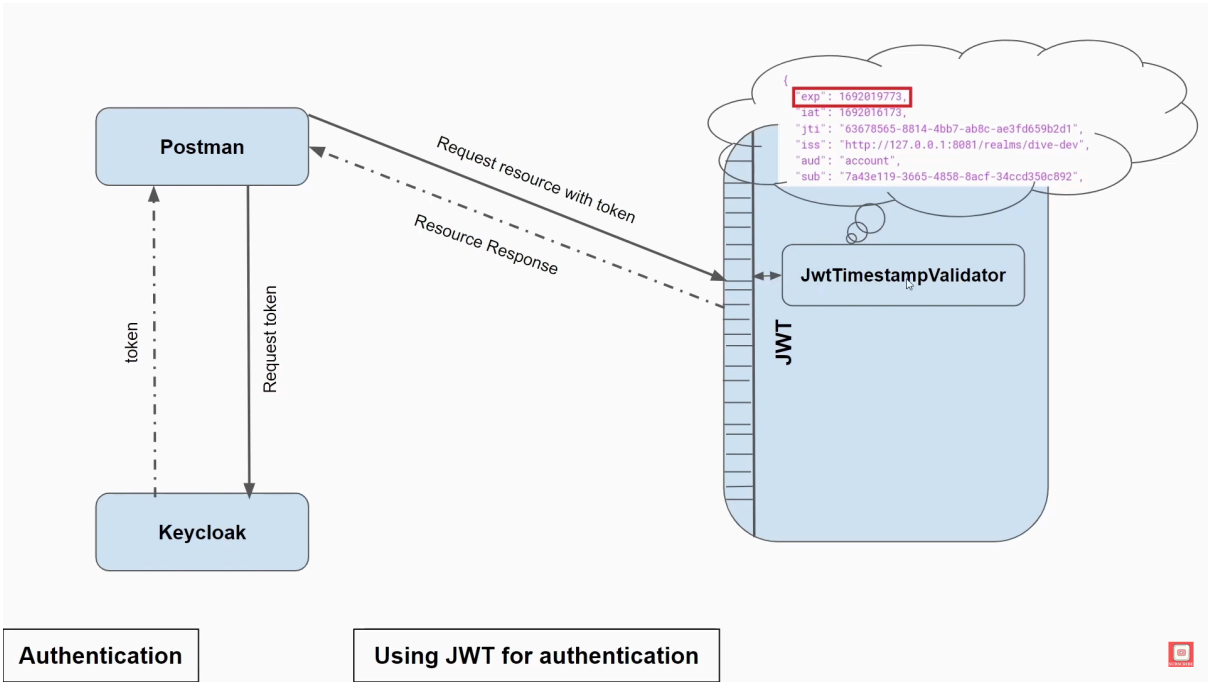
In spring boot Security use Oauth2 Resource and use JWT Token or OPAQUE Token, but in Spring security and Oauth2 either we can use JWT or OPAQUE Token,We can't use Both.

In JWT Token:

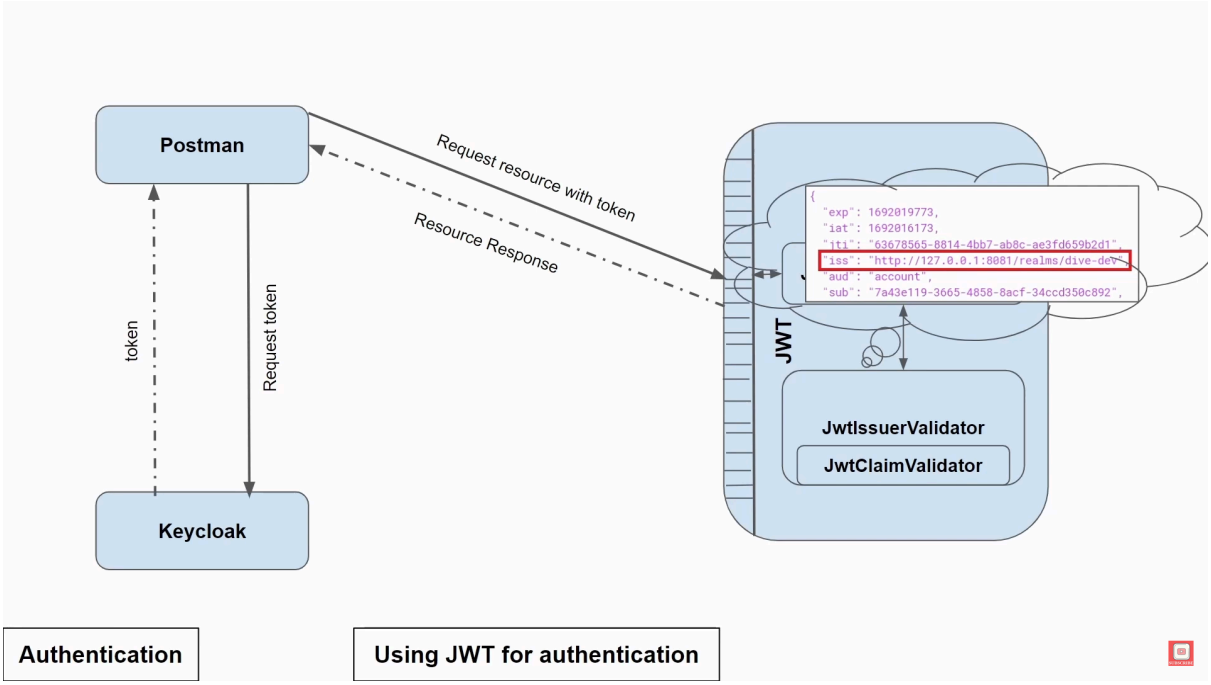
In Spring security and Oauth2 Resource JWT Token is Validated within Resource Server, It will not be sent to the Keycloak Authentication Server.



IN JWTTimestampValidator it used to check the exp date and validate.



JWTIssuerValidator will check the iss value issued by Keycloak.



But it will not Check the session is Logout for Keycloak or not so the data can be passed. This is a disadvantage in JWT Token.

Authentication:

In Spring Boot :

– IN properties file :

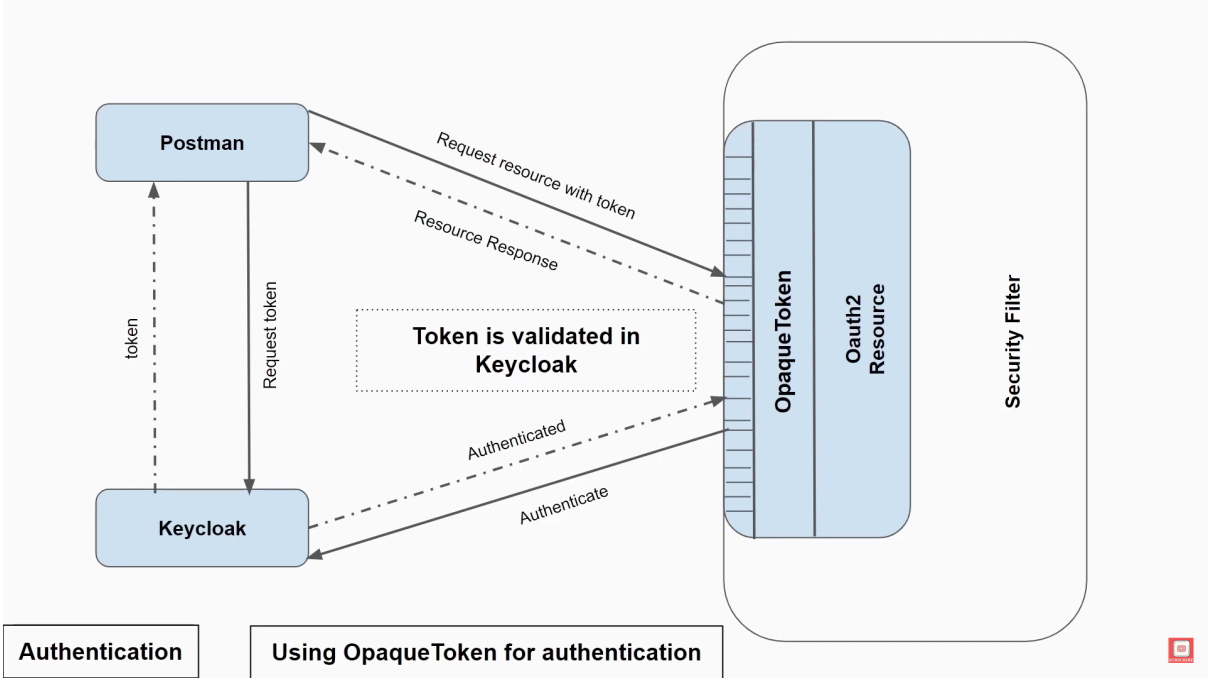
```
spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8081/realms/springboot-be
```

– IN security :

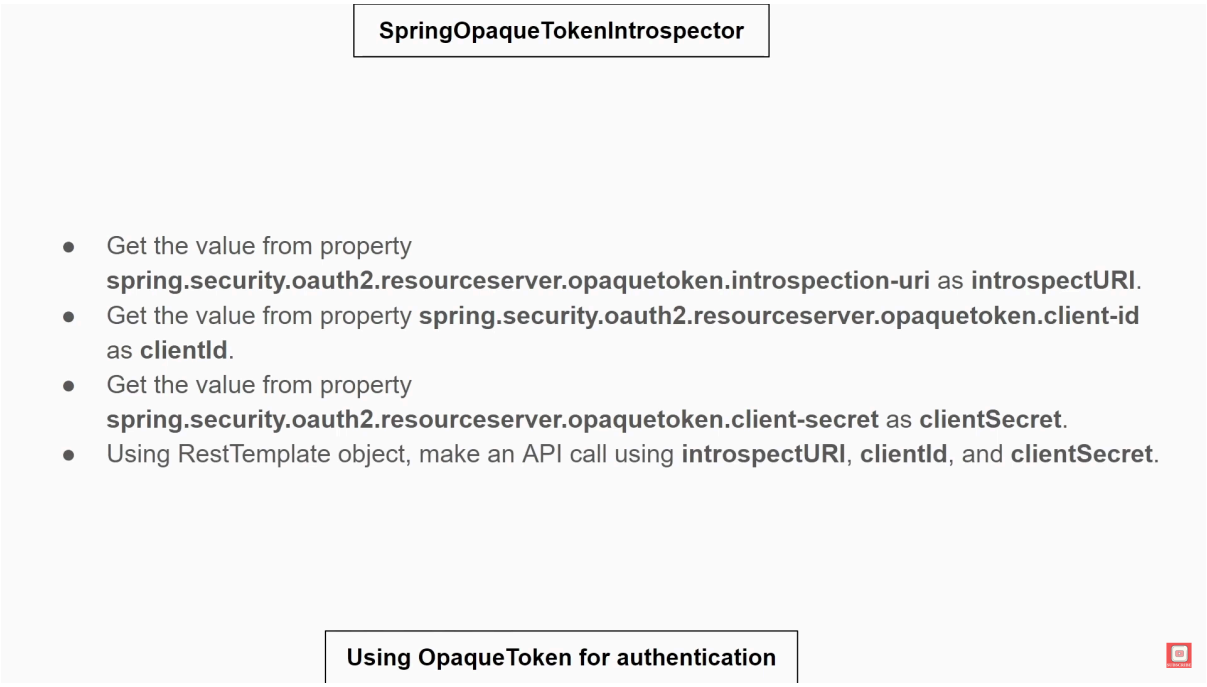
```
.oauth2ResourceServer(oauth2ResourceServerCustomizer ->
oauth2ResourceServerCustomizer.jwt(Customizer.withDefaults()));
```

OPAQUE TOKEN:

In Spring security and Oauth2 Resource OPAQUE Token is Validated Send's to Authentication Server(Keycloak Server).



From This Three property it gives token validation to Keycloak.



In Spring Boot :

– IN properties file :

```
spring.security.oauth2.resourceserver.opaquetoken.client-id=springboot-dev
spring.security.oauth2.resourceserver.opaquetoken.client-secret=7P1jVrSko0F9zPDc0tzEIxRCp9ReyzW7
spring.security.oauth2.resourceserver.opaquetoken.introspection-uri=http://localhost:8081/realms/springboot-be/protocol/openid-connect/token/introspect
```

– IN security :

```
.oauth2ResourceServer(oauth2ResourceServerCustomizer ->
oauth2ResourceServerCustomizer.opaqueToken(Customizer.withDefaults()));
```

Authorization:

In Authorization it will not send to Keycloak Server for JWT and OPAQUE Token. Adjusted the JWT Token.

Creating New CLAIM in JWT Token in KeyCloak :

JWT - Method Level Authorization

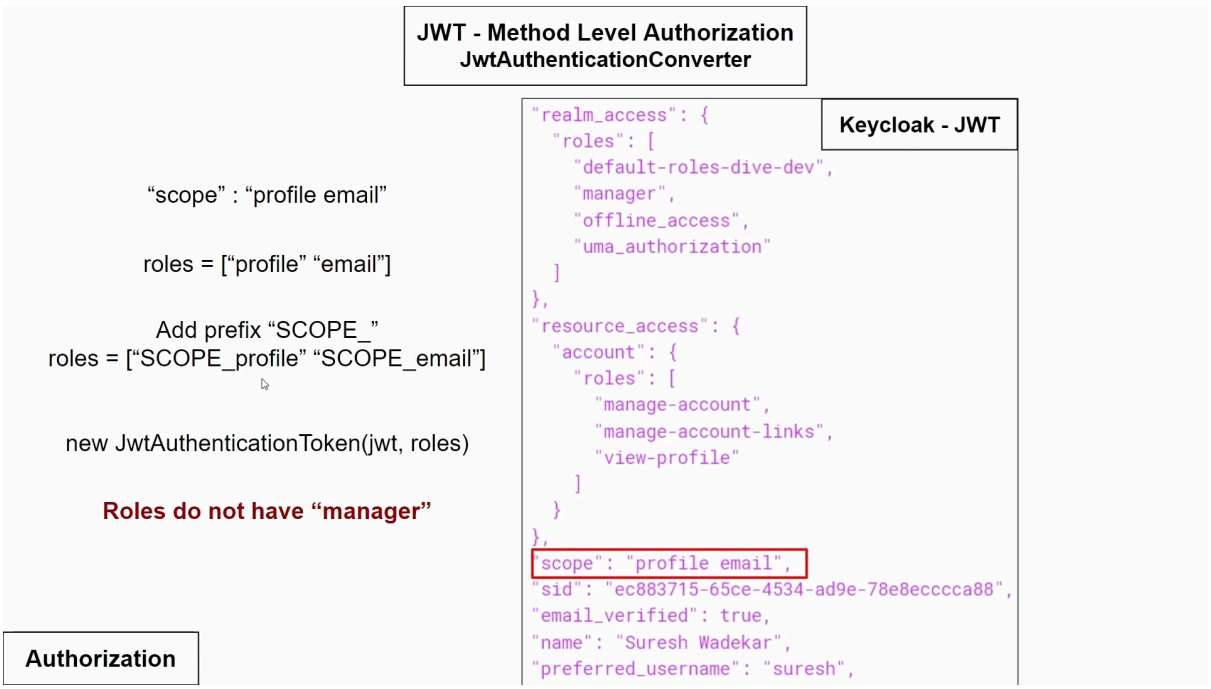
Incoming JWT token is adjusted to Spring Boot Oauth2 format


```
"realm_access": {
  "roles": [
    "default-roles-dive-dev",
    "manager",
    "offline_access",
    "uma_authorization"
  ]
},
"resource_access": {
  "account": {
    "roles": [
      "manage-account",
      "manage-account-links",
      "view-profile"
    ]
  }
},
"scope": "profile email",
"sid": "ec883715-65ce-4534-ad9e-78e8ecccca88",
"email_verified": true,
"name": "Suresh Wadekar",
"preferred_username": "suresh",
```


JWT - Method Level Authorization
JwtAuthenticationConverter

- Check if token contains property “scope” or “scp” in jwt access token.
  - Checks for “scp” only if “scope” is unavailable.
- Append the prefix “SCOPE_” to each entry in the value of the property.
  - Works on only the String value having roles separated by space.
  - Or a string array of roles.
- Adds these appended entries as roles in JwtAuthenticationToken class.
- These are then used to authorize each API accordingly.

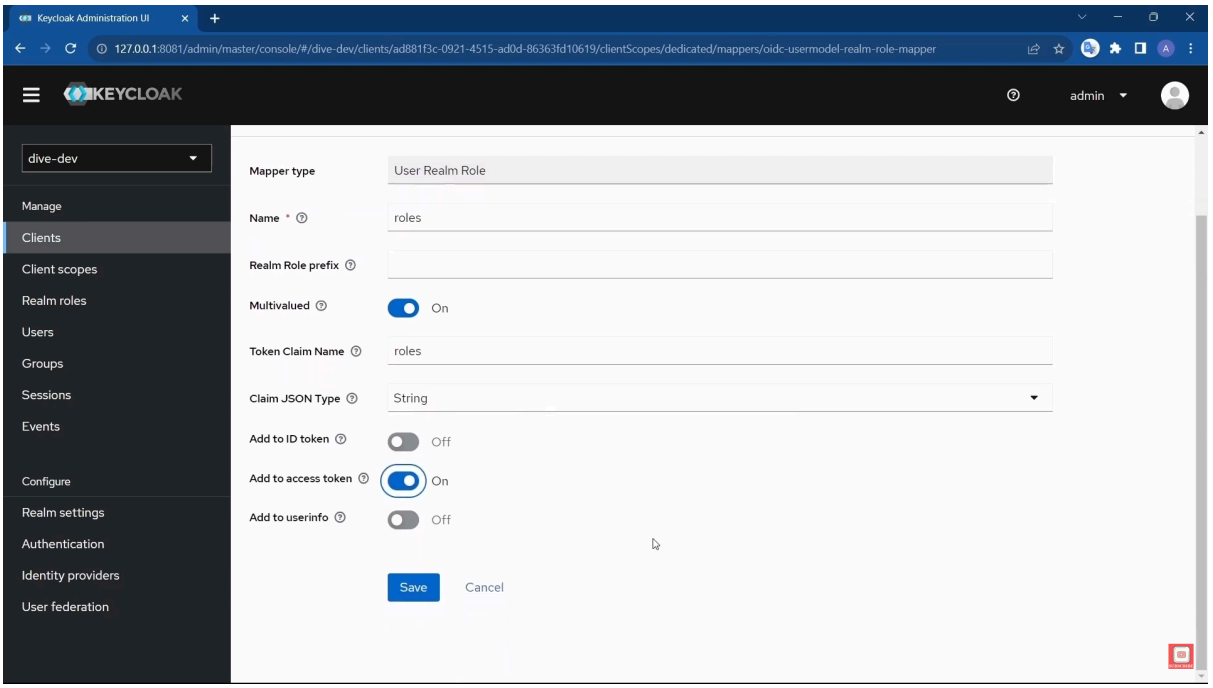
Authorization



Creating New Data Inside KeyCloak:

Adding Role:

Insert Client->client Scope -> default client scope (springboot-dev-dedicated) -> configure new mapper -> user Realm Role



And get roles by getting directly with the JWT Token. And Enable the Swagger in springdoc (springdoc) is directly access by spring boot 3 so we can make less amount of implementation

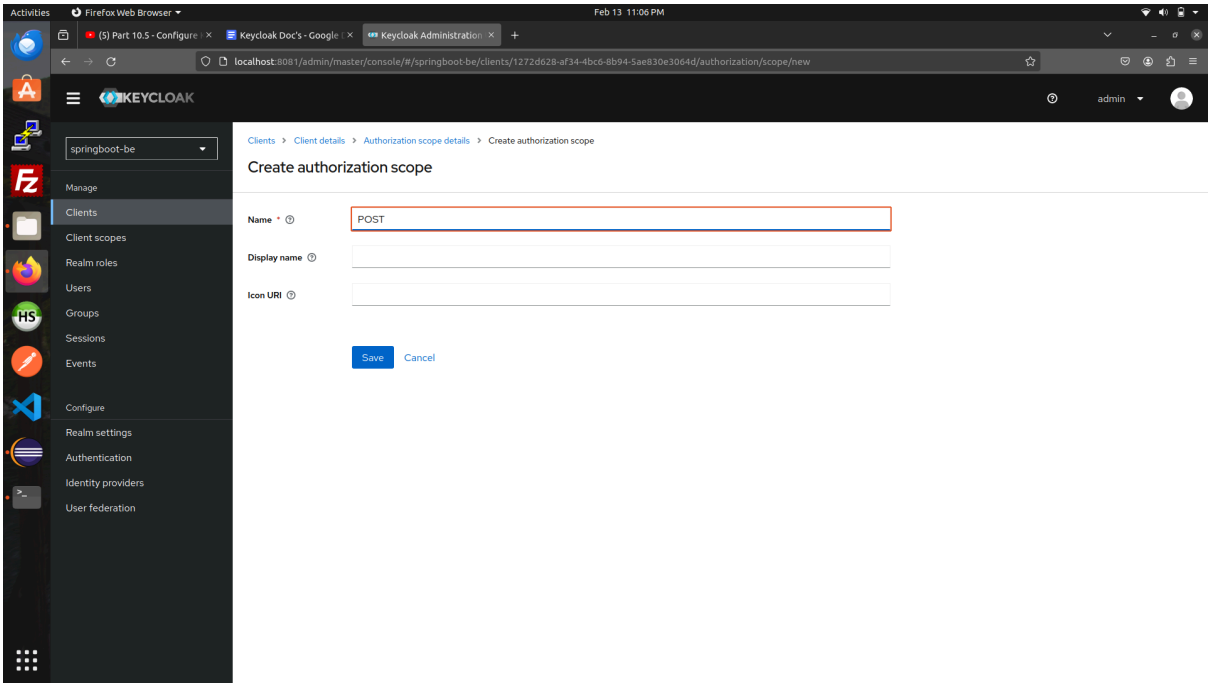
Ref : https://springdoc.org/ and https://reflectoring.io/spring-boot-springdoc/

Authentication:

Enable the **Authentication** so It will be enabled.
You will see the Authentication Table,go to the Tab and Delete the Resource, Policies if you delete the Policies Permission will be deleted.

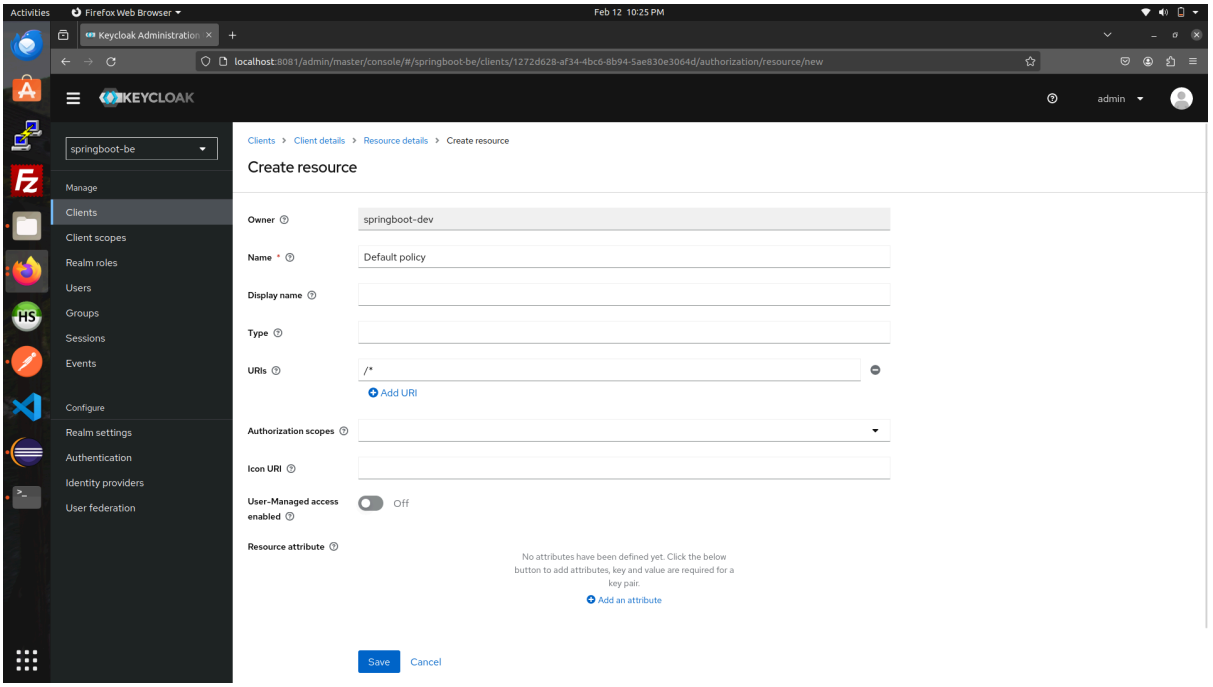
Create a new Resource,policies,scope and Permission

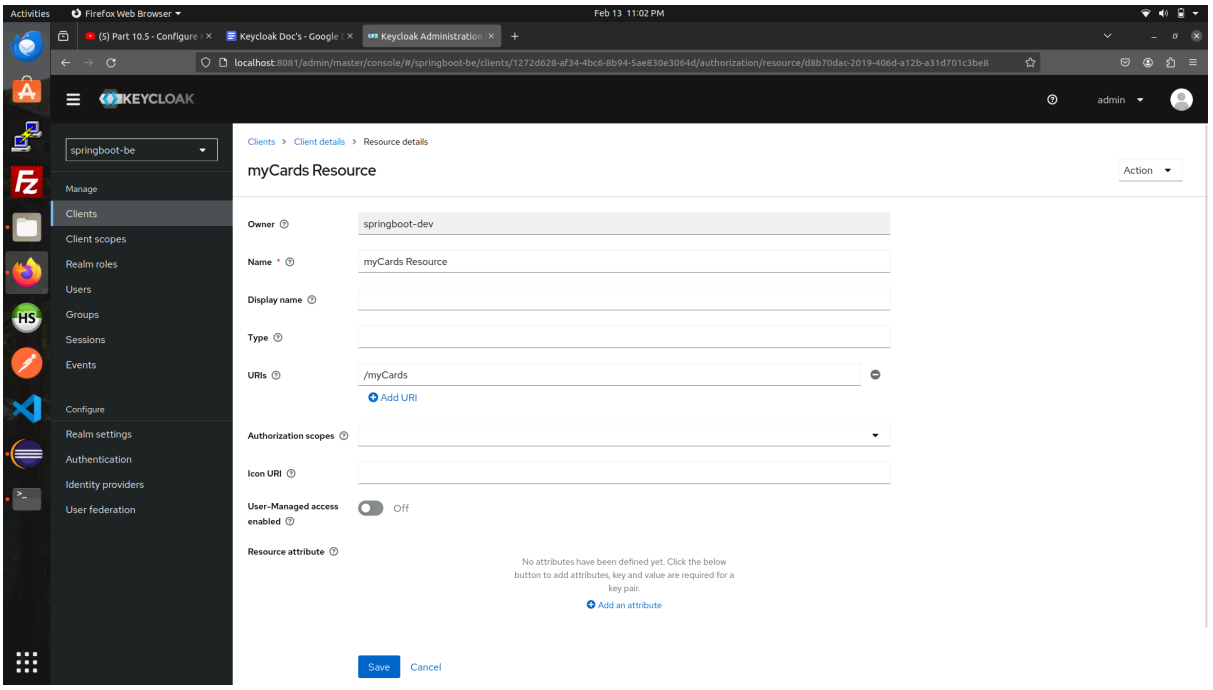
In Scope:
In Scope (we can define POST,GET,PUT methods) we can configure resources.
If we want to use Scope we need to put `"http-method-as-scope":true,` in JSON file in spring boot Application.



In Resource:

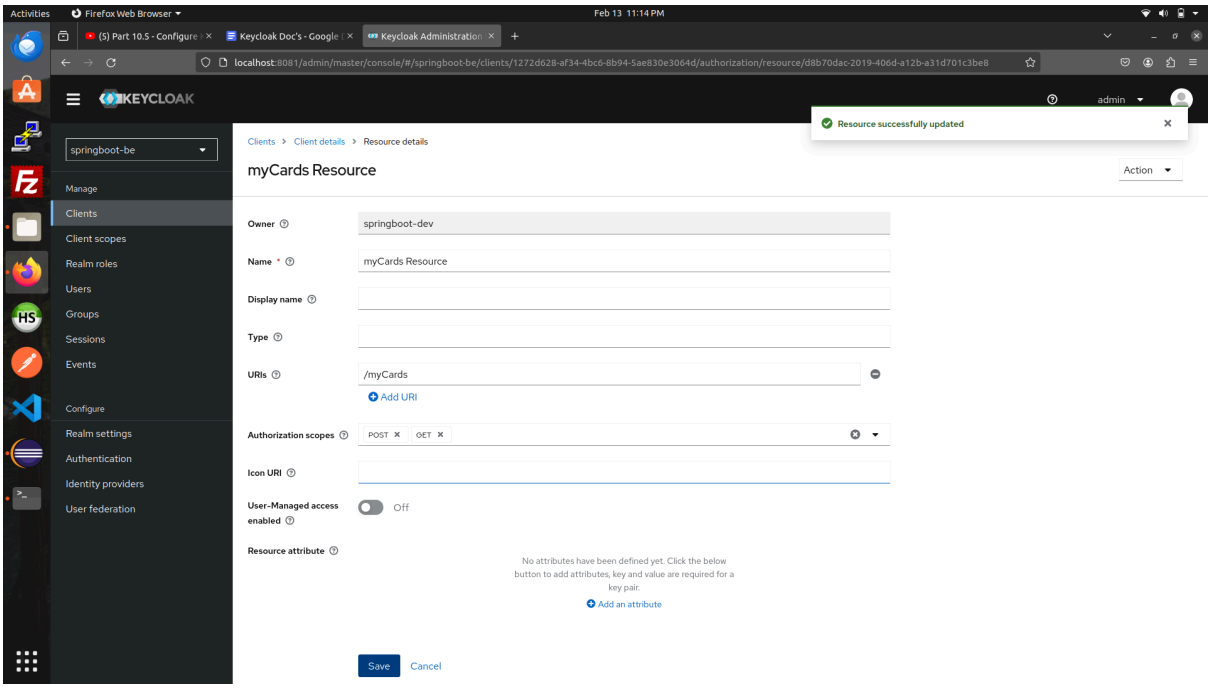
Resource without Scope :





Resource with Scope :

Resource will be one i.e API can be one but methods can be different as POST,GET etc.. So,We need to add scope in Authorization scopes.

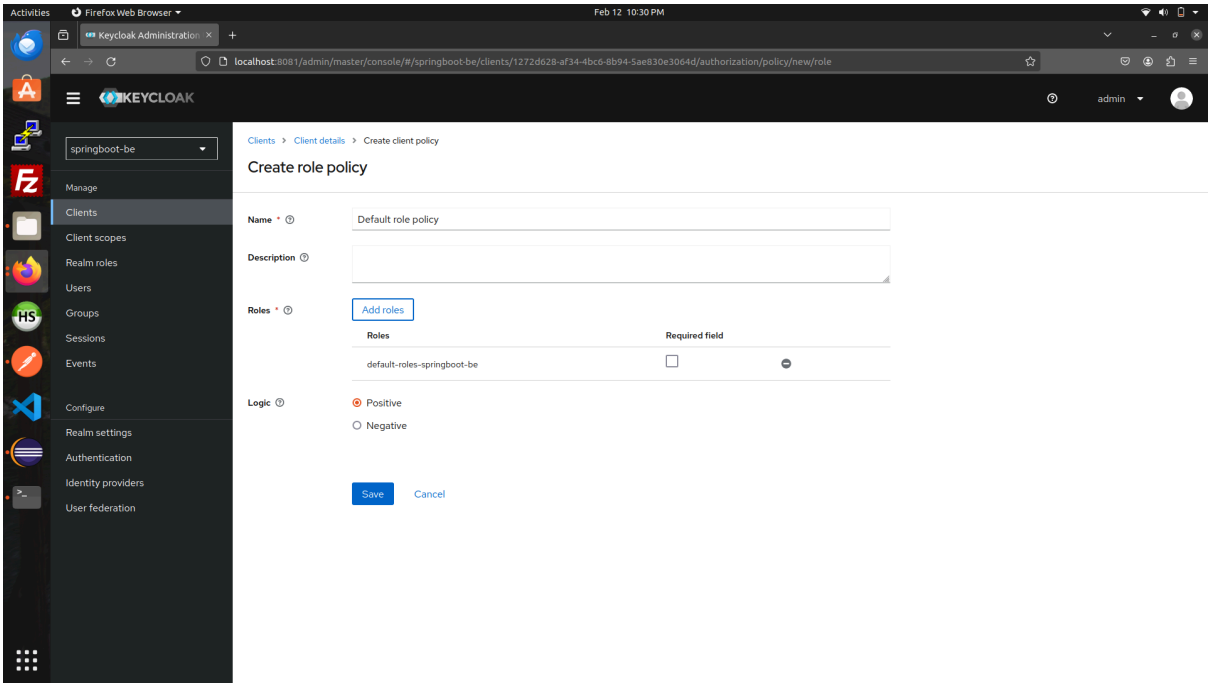


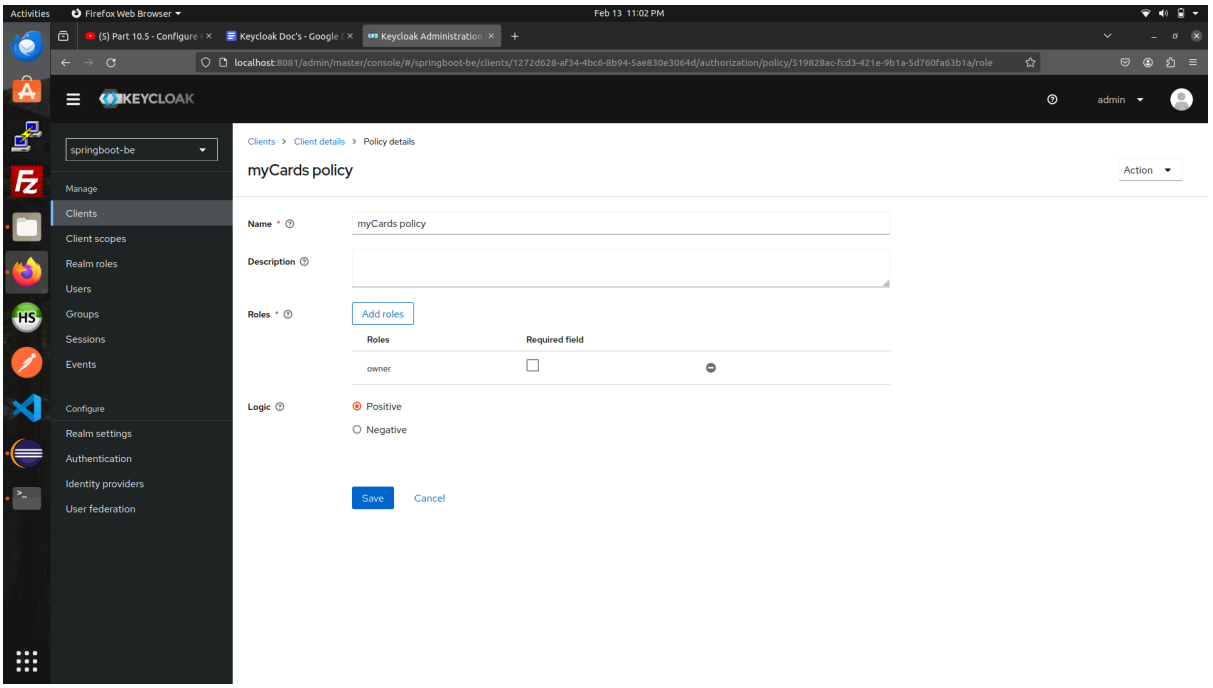
In Policies:

Create a new Policies select Role.

In Add role select (default-roles-springboot-be) because Every user has this.

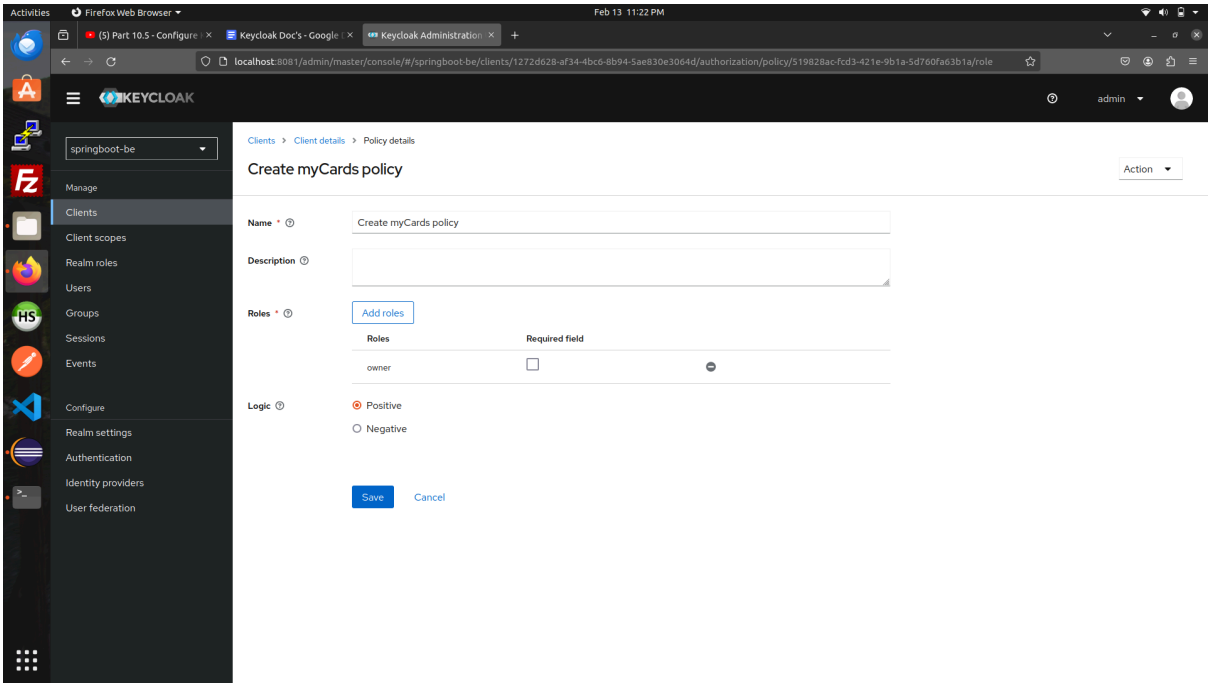
Policies without Scope :

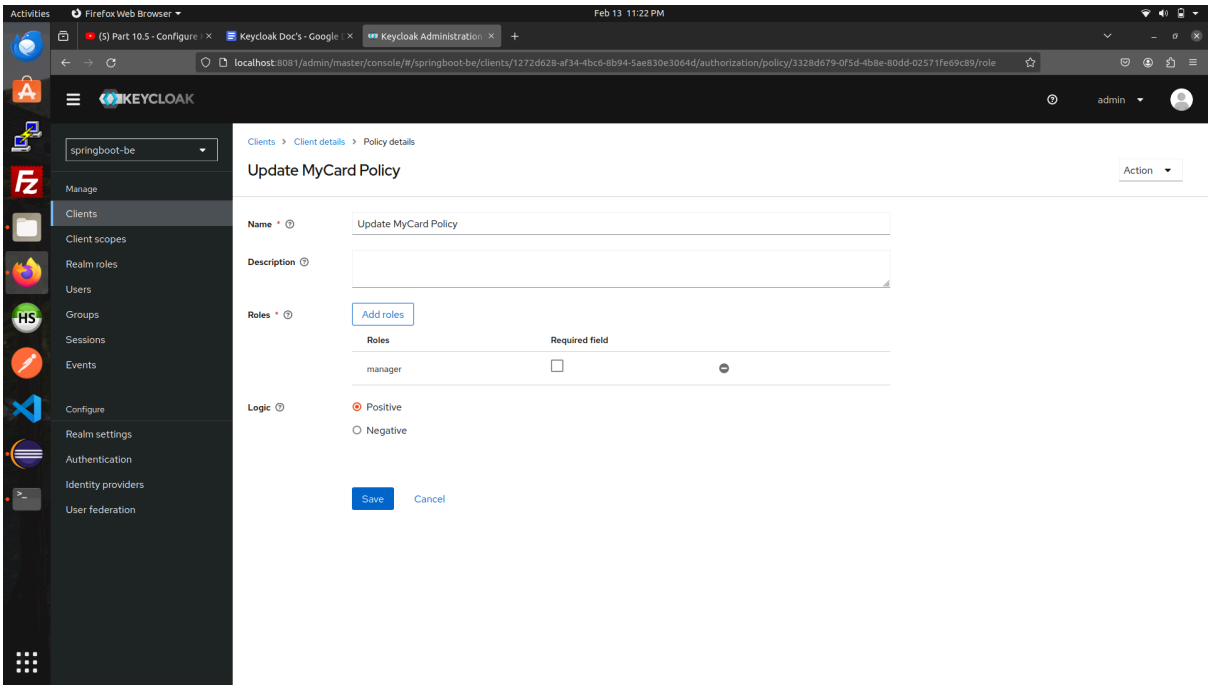




Policies with Scope :

We have to create different policies like manager, owner ect…



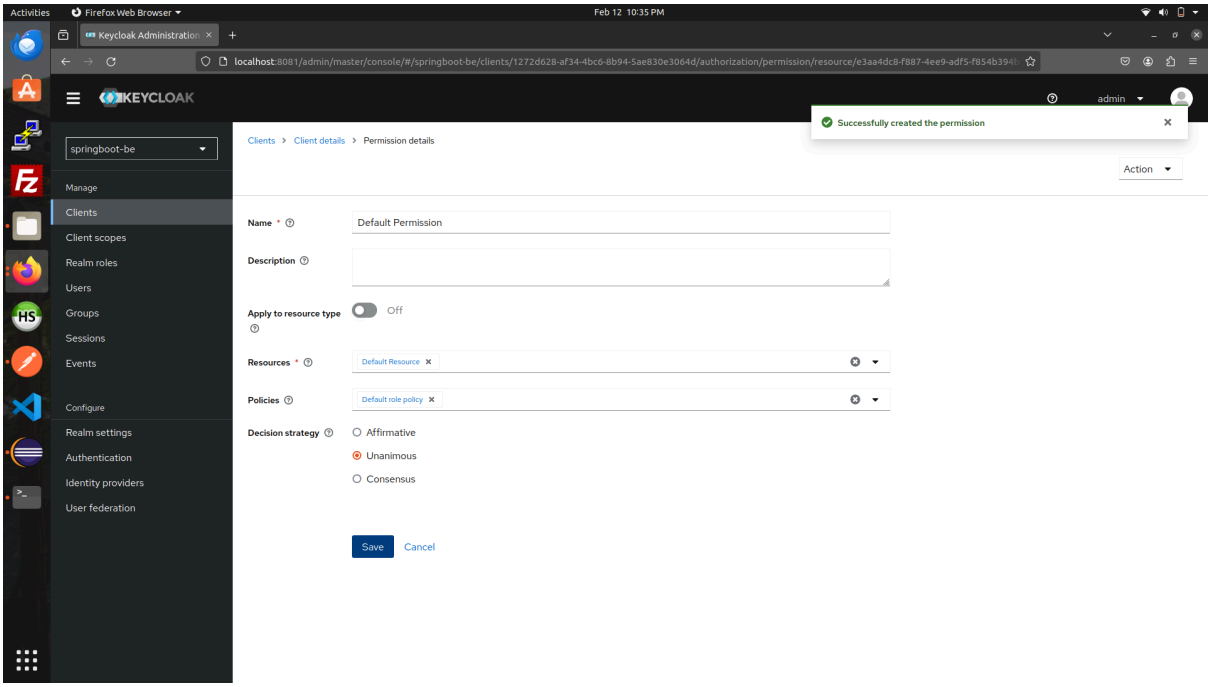


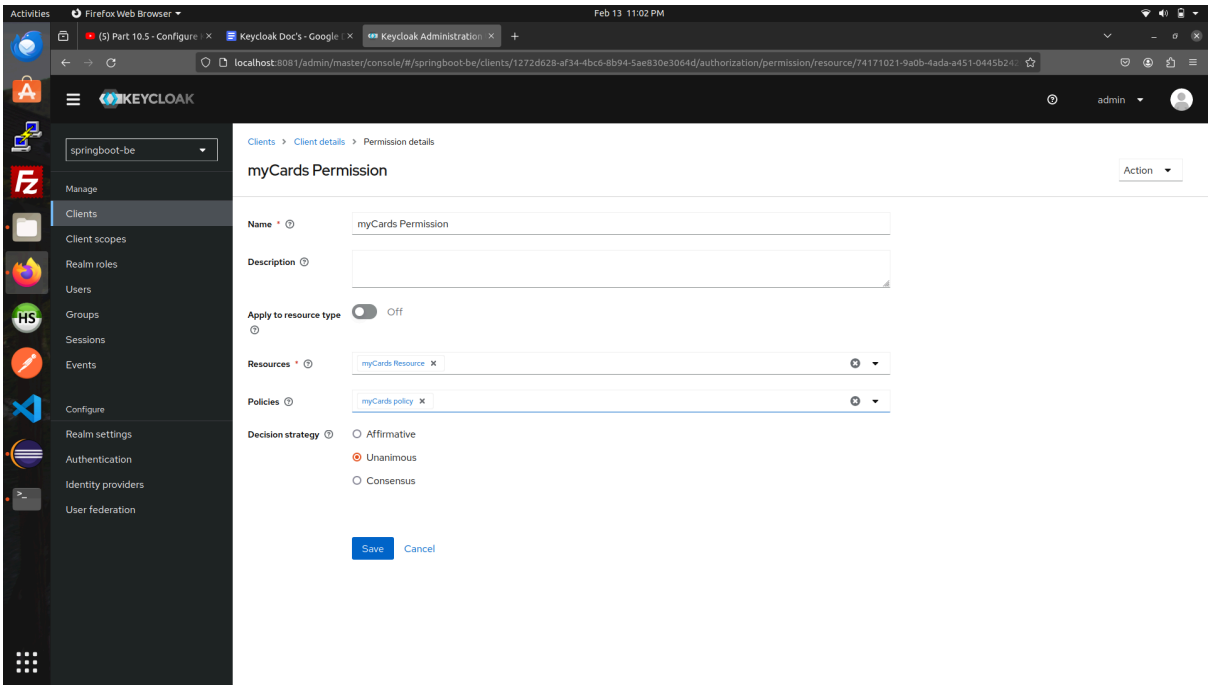
In Permission:

Two types of Permission are Resource-base Permission and Scope-base Permission.

We created Resource only Resource-base Permission is enabled.

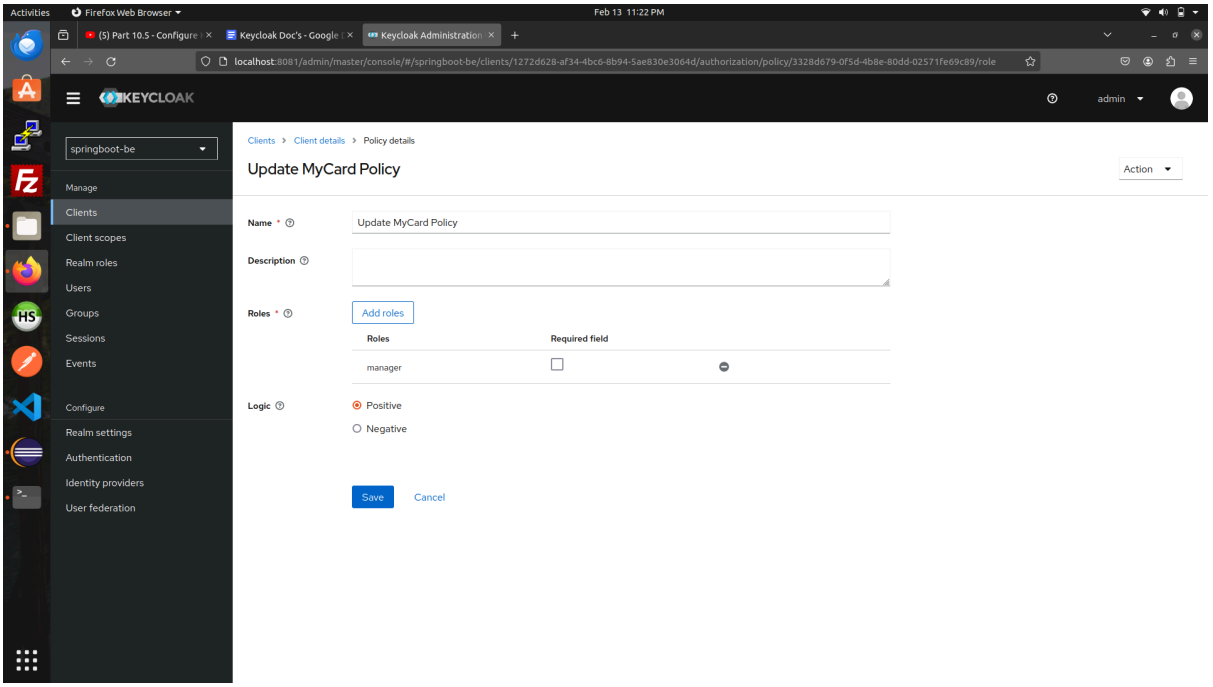
Permission without Scope :

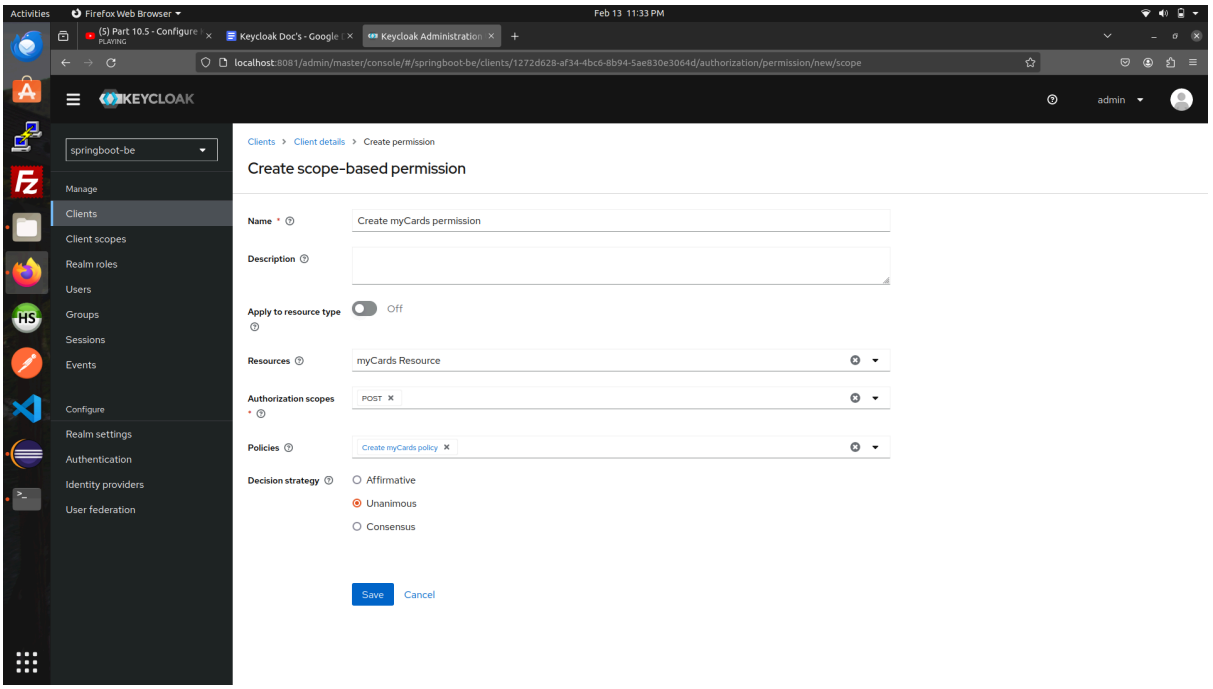




Permission with Scope :

We have to create different **Permission** like POST,GET ect…
Create permission select Add bases cope.


Activities
Firefox Web Browser
Feb 13 11:22 PM
(5) Part 10.5 - Configure
Keycloak Doc's - Google
Keycloak Administration
localhost:8081/admin/master/console/#/springboot-be/clients/1272d628-af34-4bc6-8b94-5ae830e3064d/authorization/policy/3328d679-0f5d-4b8e-80dd-02571fe69c89/role
KEYCLOAK
admin
springboot-be
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details > Policy details
Update MyCard Policy
Action
Name *
Update MyCard Policy
Description
Roles *
Add roles
Roles
Required field
manager
Logic
Positive
Negative
Save
Cancel


Activities
Firefox Web Browser
Feb 13 11:33 PM
(5) Part 10.5 - Configure
PLAYING
Keycloak Doc's - Google
Keycloak Administration
localhost:8081/admin/master/console/#/springboot-be/clients/1272d628-af34-4bc6-8b94-5ae830e3064d/authorization/permission/new/scope
KEYCLOAK
admin
springboot-be
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details > Create permission
Create scope-based permission
Name *
Create myCards permission
Description
Apply to resource type
Off
Resources
myCards Resource
Authorization scopes *
POST
Policies
Create myCards policy
Decision strategy
Affirmative
Unanimous
Consensus
Save
Cancel

In Authentication:

Different Type of Policies:

Role -> Role base **Authentication**

User -> User base **Authentication**

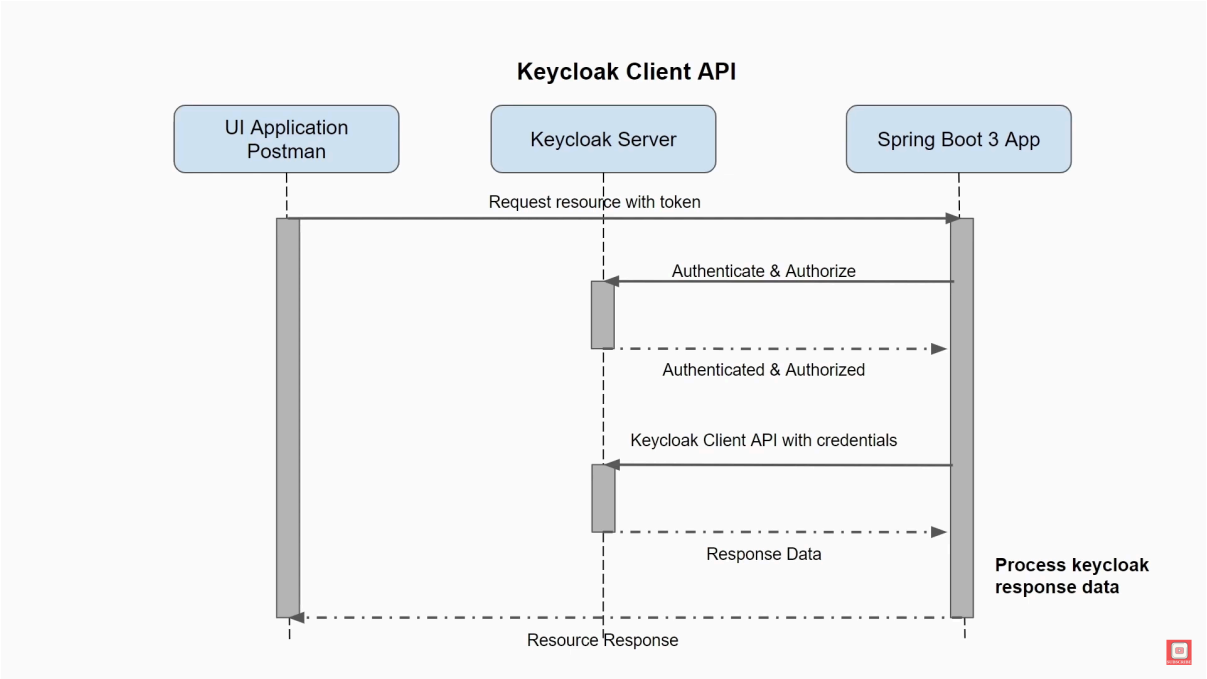
Group -> If we have lot of user for one API

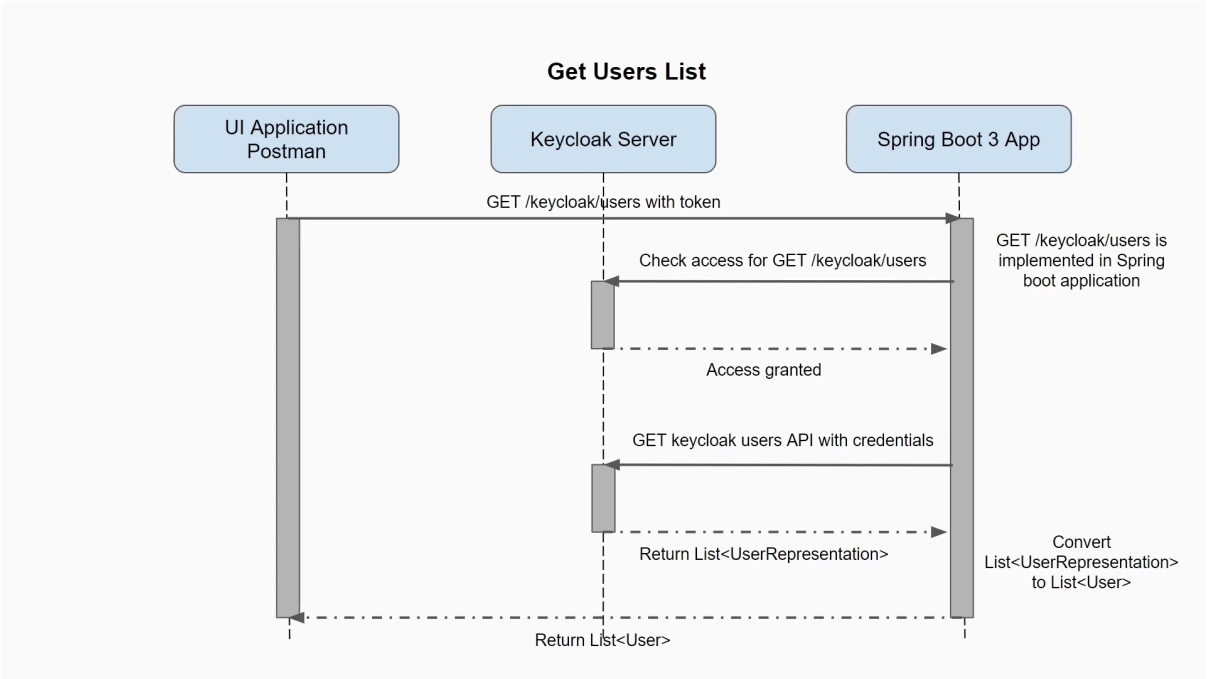
eg: /mycard has to access different type of user need to access this API we can create Group and add the user and we can also add child group so the all user can access this API

Time -> we can set time for API so that time only API can access. And it work

Regex -> It works on regex patterns,for details see YOUTUBE videos.

In springBoot:



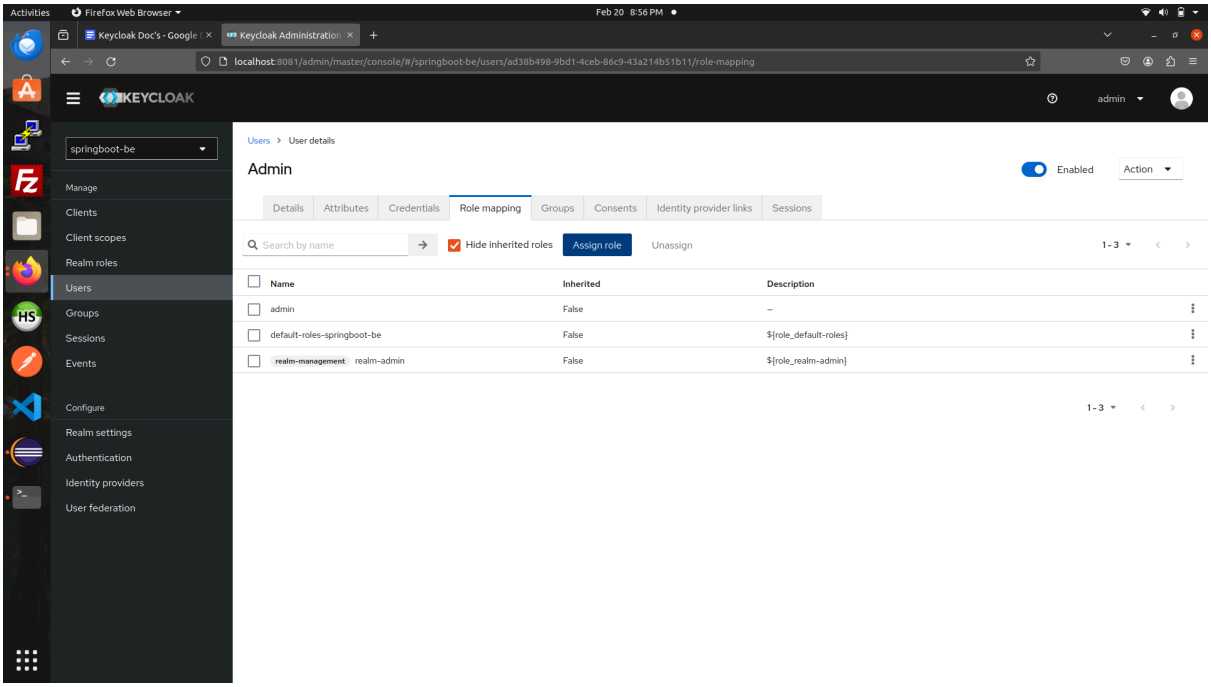


Create Admin as User:

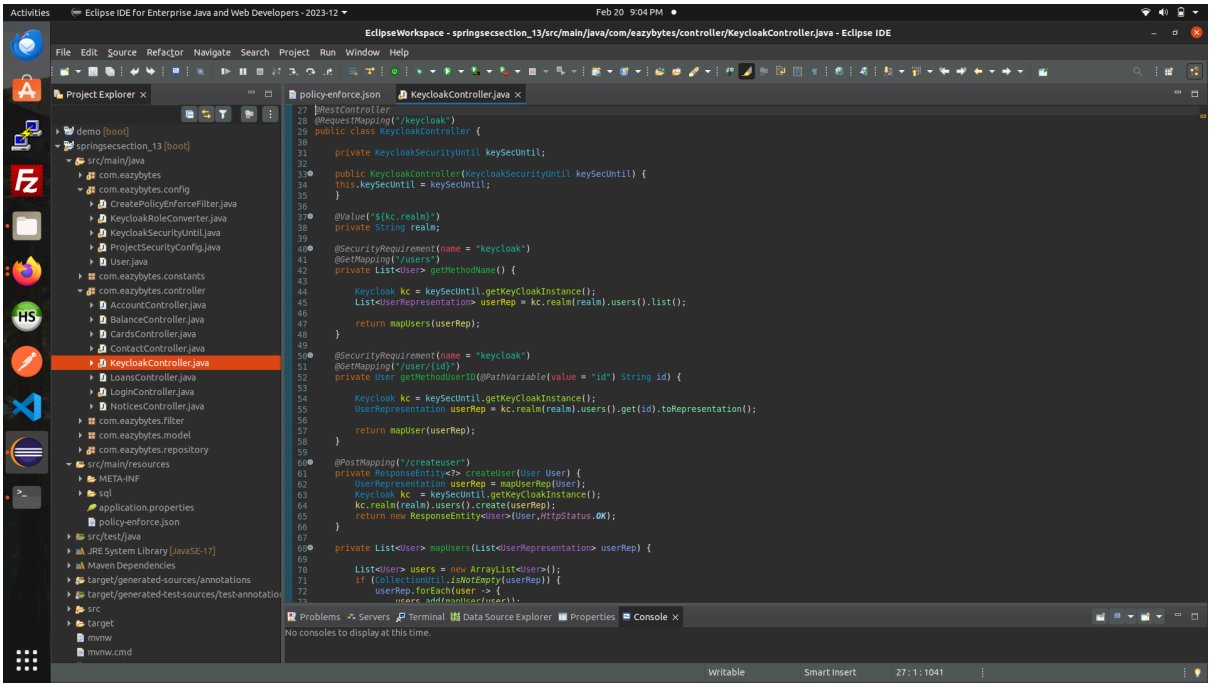
Need to create admin and password and role as ADMIN

In Role mapping Click Assign Role -> Filter by realm role -> Filter by Client -> search for realm-admin.

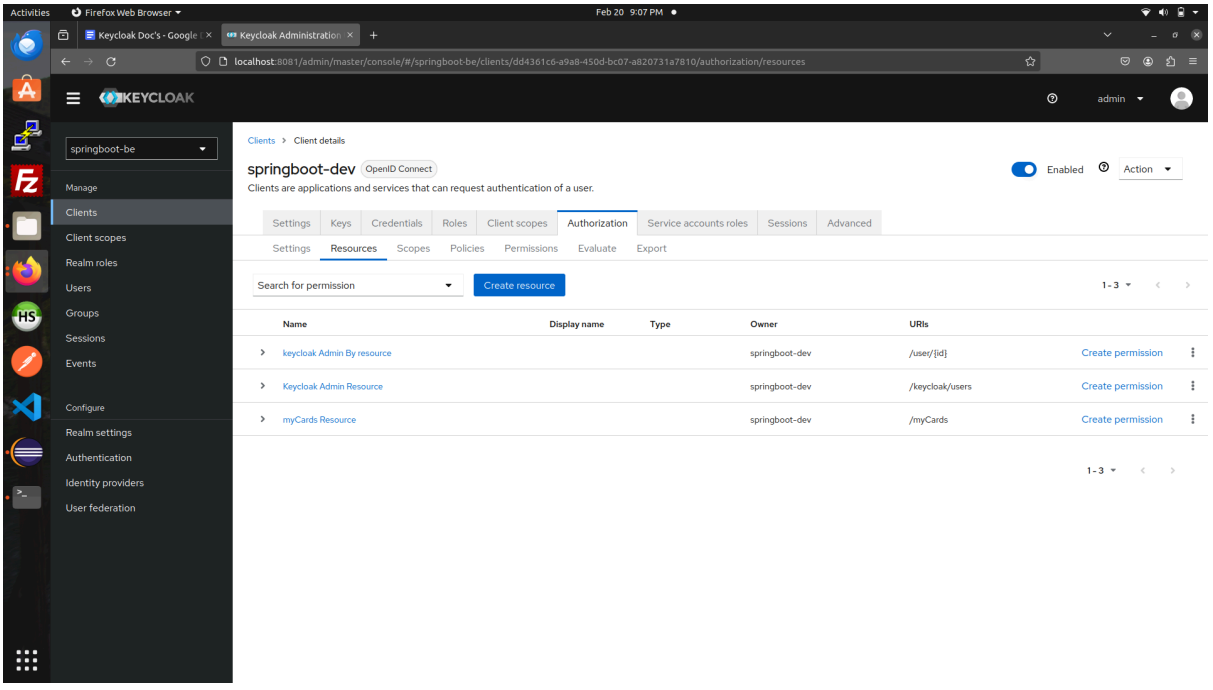
Add Role as ADMIN.

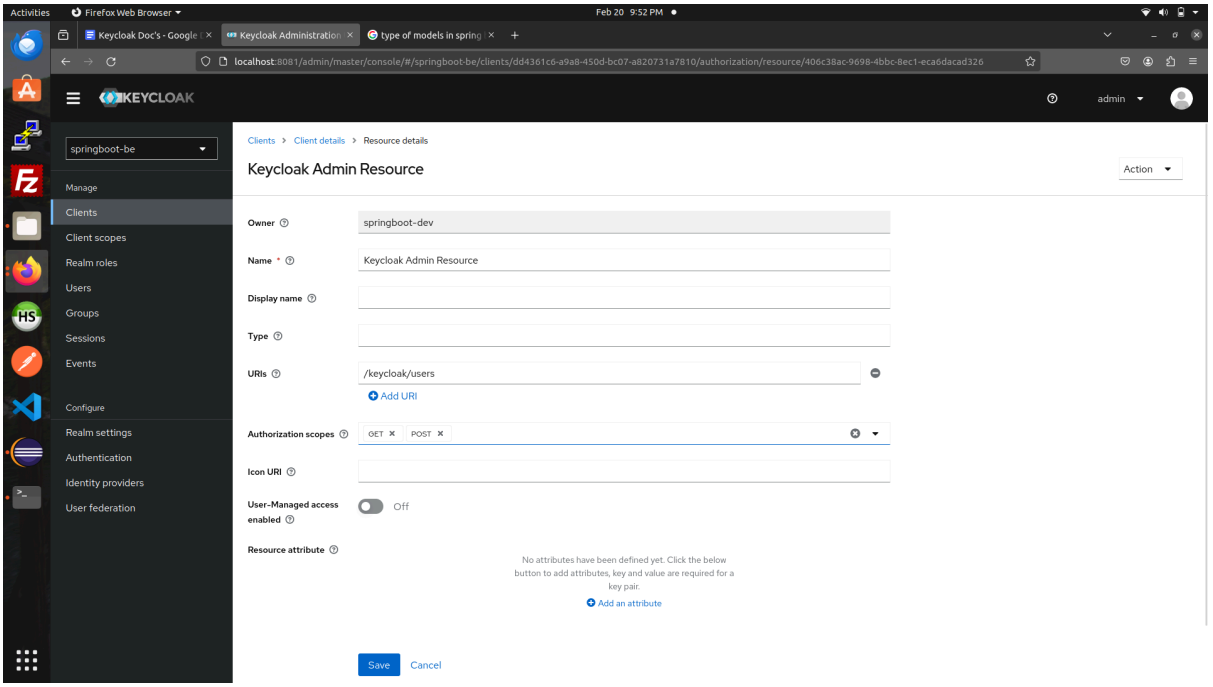


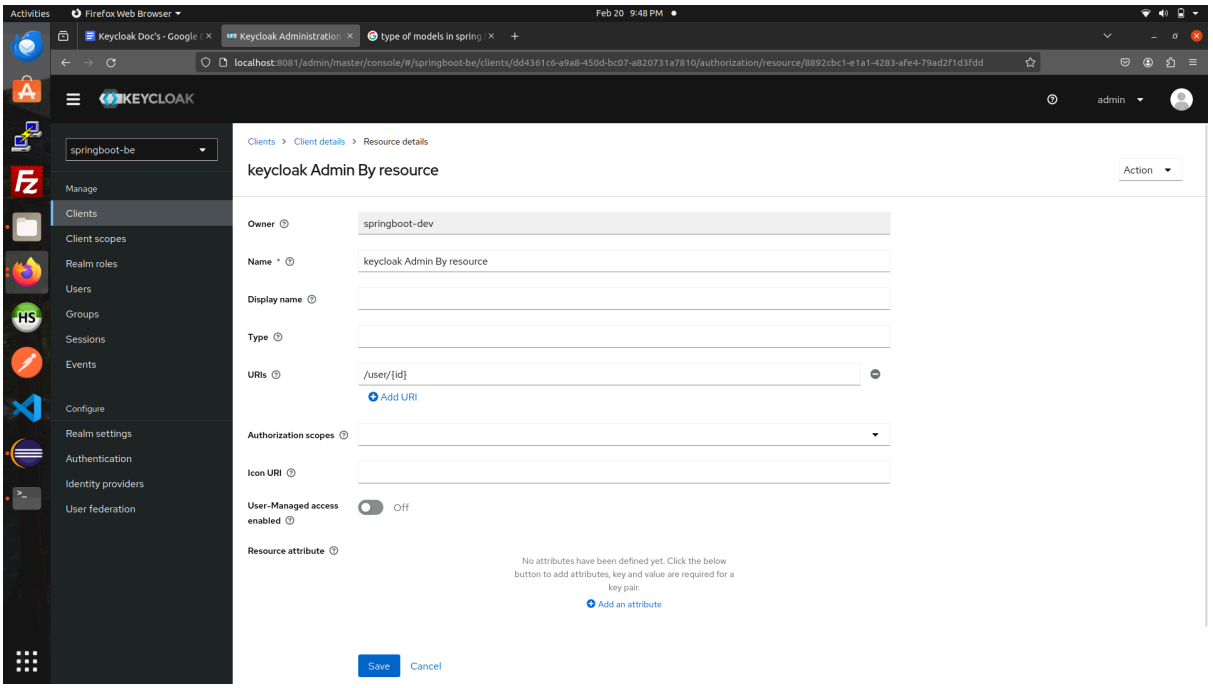
Need to create Resource,Scope,Policy and Permission for /keycloak API's

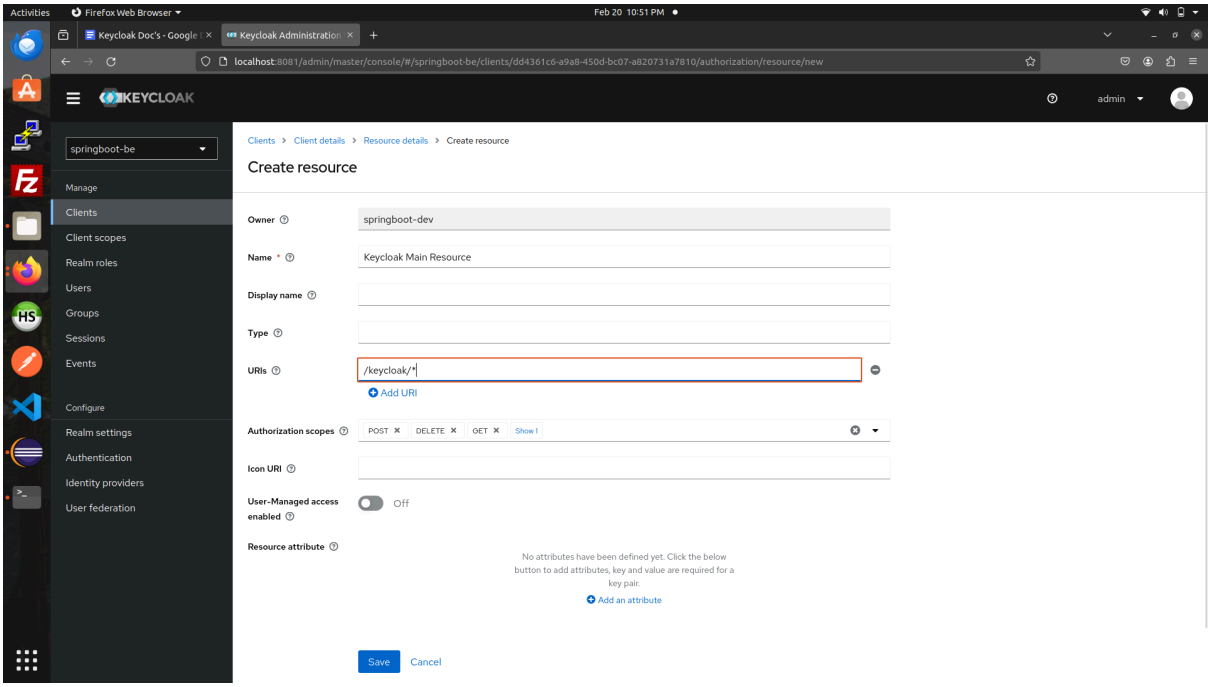


Create Resource for /keycloak API:



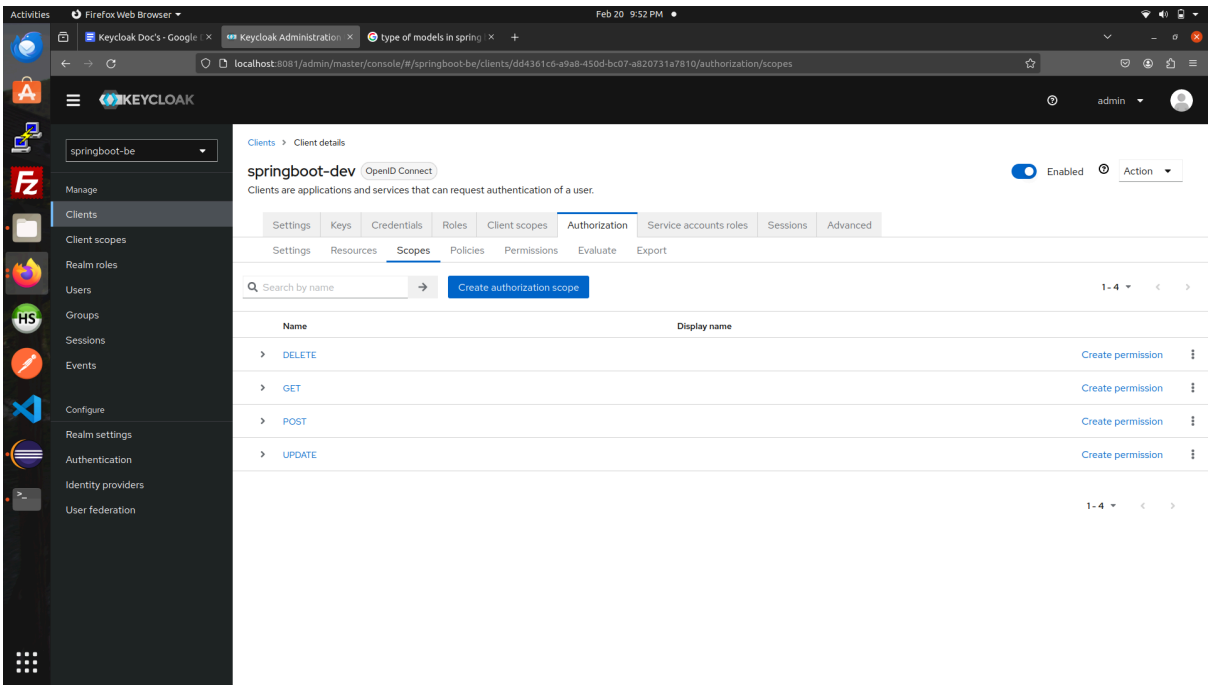

Activities
Firefox Web Browser
Feb 20 9:52 PM
Keycloak Doc's - Google
Keycloak Administration
type of models in spring
localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/resource/406c38ac-9698-4bbc-8ec1-eca6dacad326
KEYCLOAK
admin
springboot-be
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details > Resource details
Keycloak Admin Resource
Action
Owner
springboot-dev
Name *
Keycloak Admin Resource
Display name
Type
URIs
/keycloak/users
Add URI
Authorization scopes
GET
POST
Icon URI
User-Managed access enabled
Off
Resource attribute
No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair.
Add an attribute
Save
Cancel


Activities
Firefox Web Browser
Feb 20 9:48 PM
Keycloak Doc's - Google
Keycloak Administration
type of models in spring
localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/resource/8892cbc1-e1a1-4283-afe4-79ad2f1d3fdd
KEYCLOAK
admin
springboot-be
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details > Resource details
keycloak Admin By resource
Action
Owner
springboot-dev
Name *
keycloak Admin By resource
Display name
Type
URIs
/user/{id}
Add URI
Authorization scopes
Icon URI
User-Managed access enabled
Off
Resource attribute
No attributes have been defined yet. Click the below button to add attributes, key and value are required for a key pair.
Add an attribute
Save
Cancel

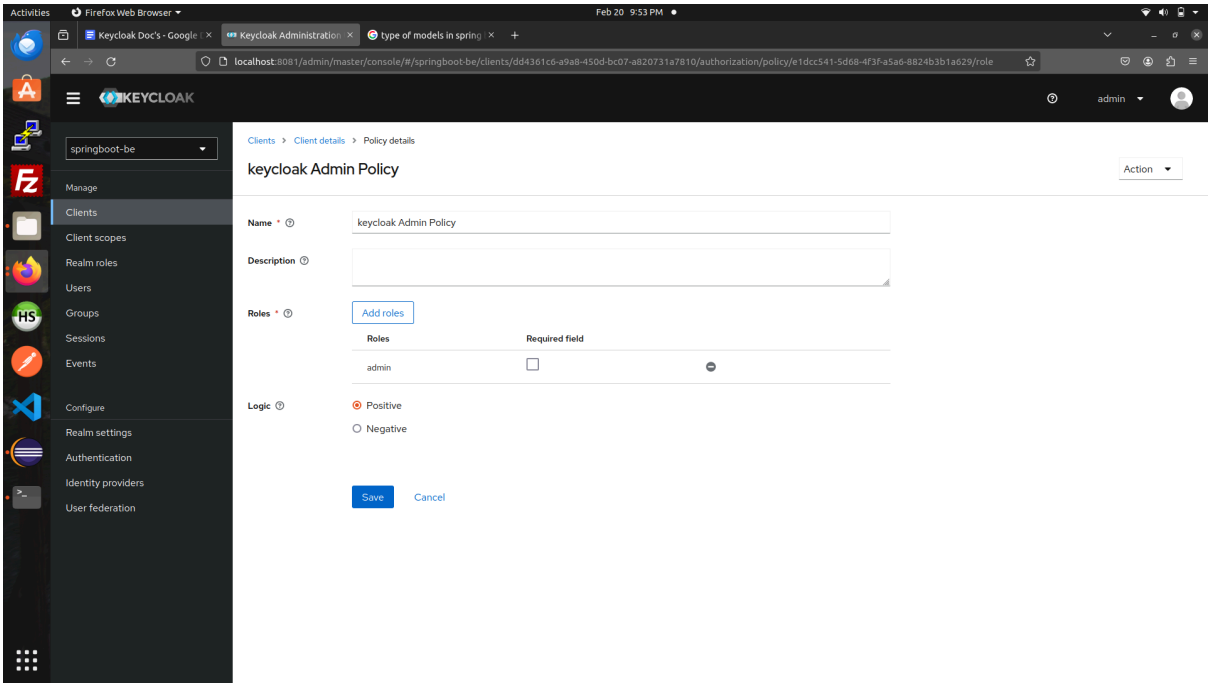


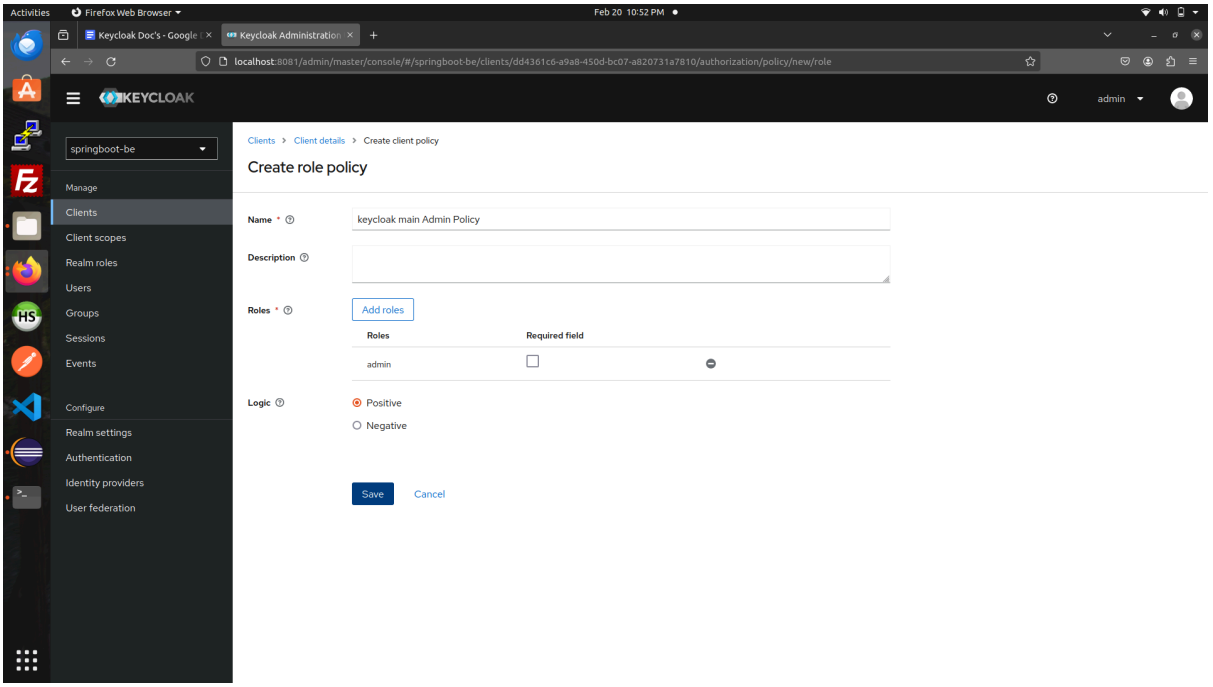
Create Scope for /keycloak API:

Create Scope as POST,GET,PUT (or) UPDATE and DELETE.



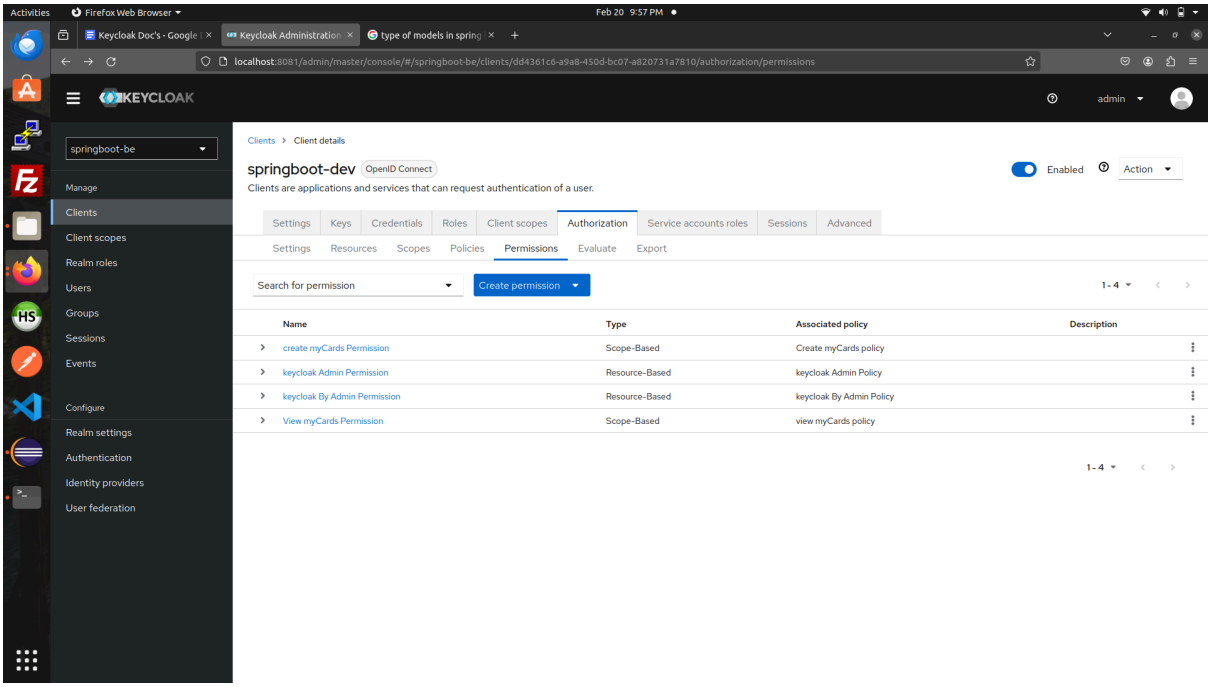
Create Policy For /keycloak API:

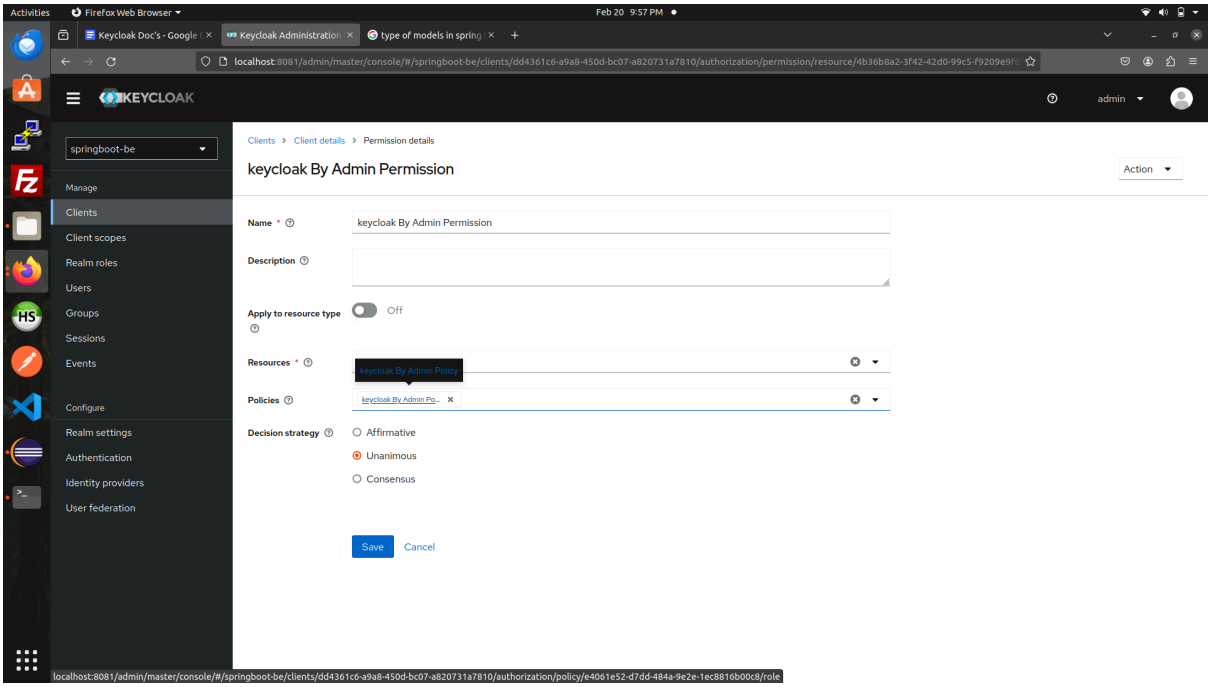


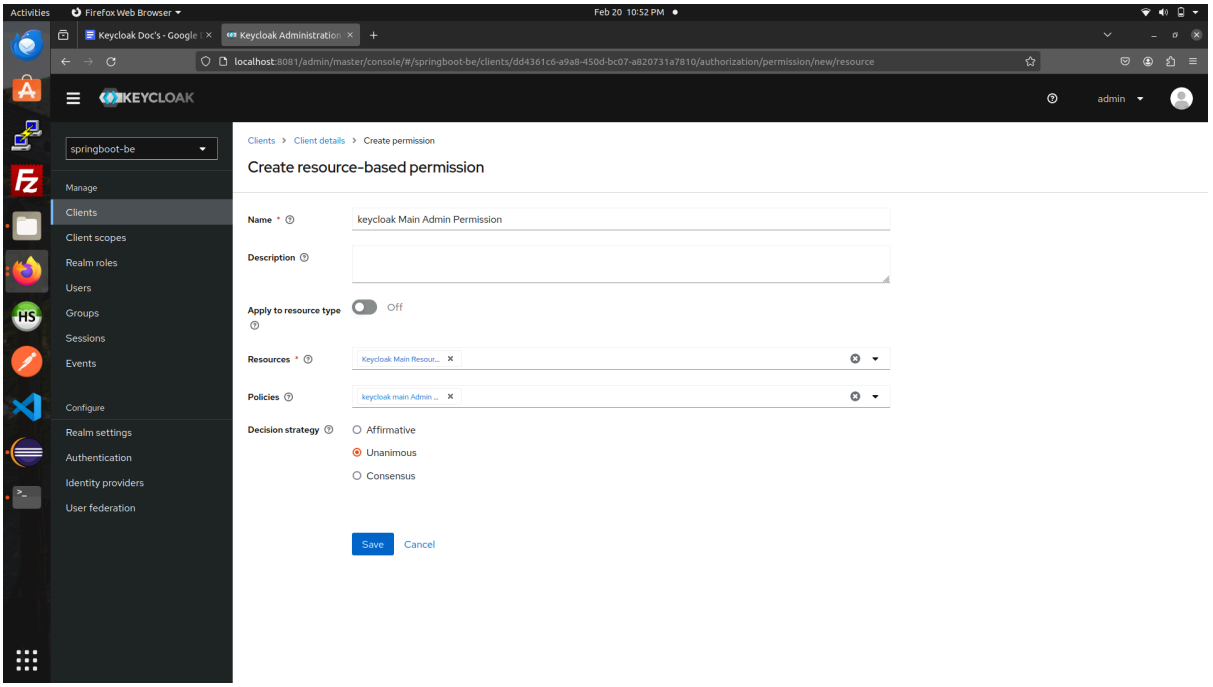


Create Permissionfor /keycloak API:

While creating permission click Resource Base.

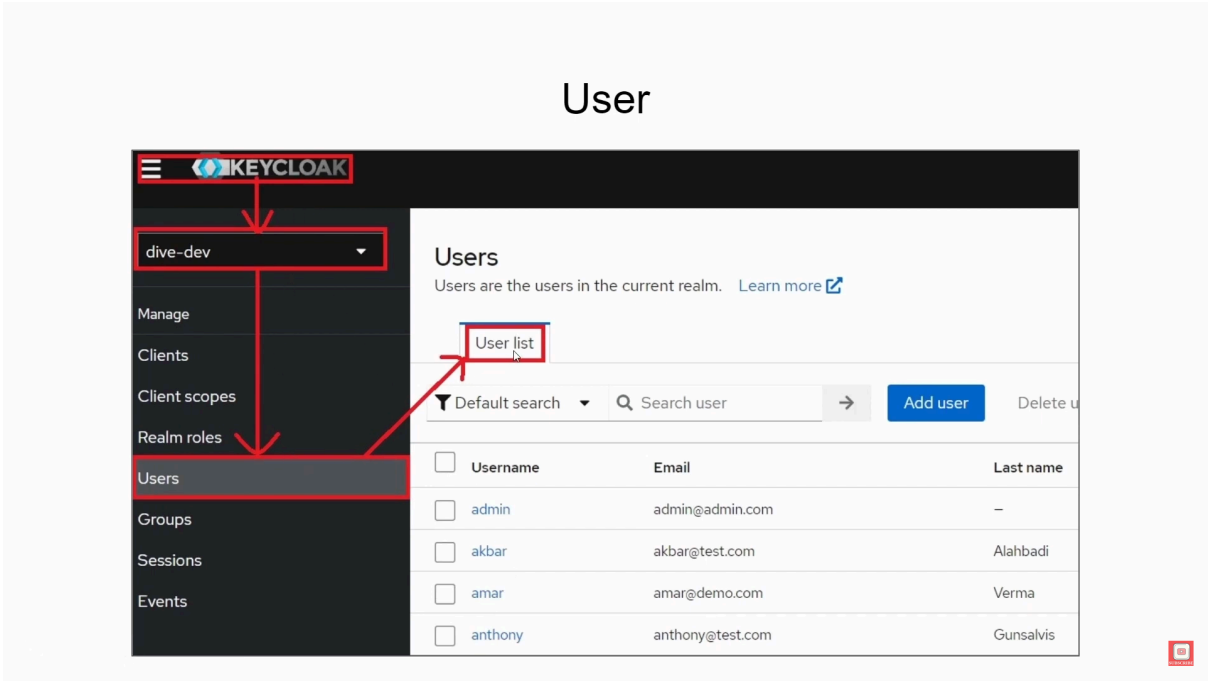

Activities
Firefox Web Browser
Feb 20 9:57 PM
Keycloak Doc's - Google
Keycloak Administration
type of models in spring
localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/permissions
KEYCLOAK
admin
springboot-be
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details
springboot-dev
OpenID Connect
Clients are applications and services that can request authentication of a user.
Enabled
Action
Settings
Keys
Credentials
Roles
Client scopes
Authorization
Service accounts roles
Sessions
Advanced
Settings
Resources
Scopes
Policies
Permissions
Evaluate
Export
Search for permission
Create permission
1 - 4
Name
Type
Associated policy
Description
create myCards Permission
Scope-Based
Create myCards policy
keycloak Admin Permission
Resource-Based
keycloak Admin Policy
keycloak By Admin Permission
Resource-Based
keycloak By Admin Policy
View myCards Permission
Scope-Based
view myCards policy
1 - 4

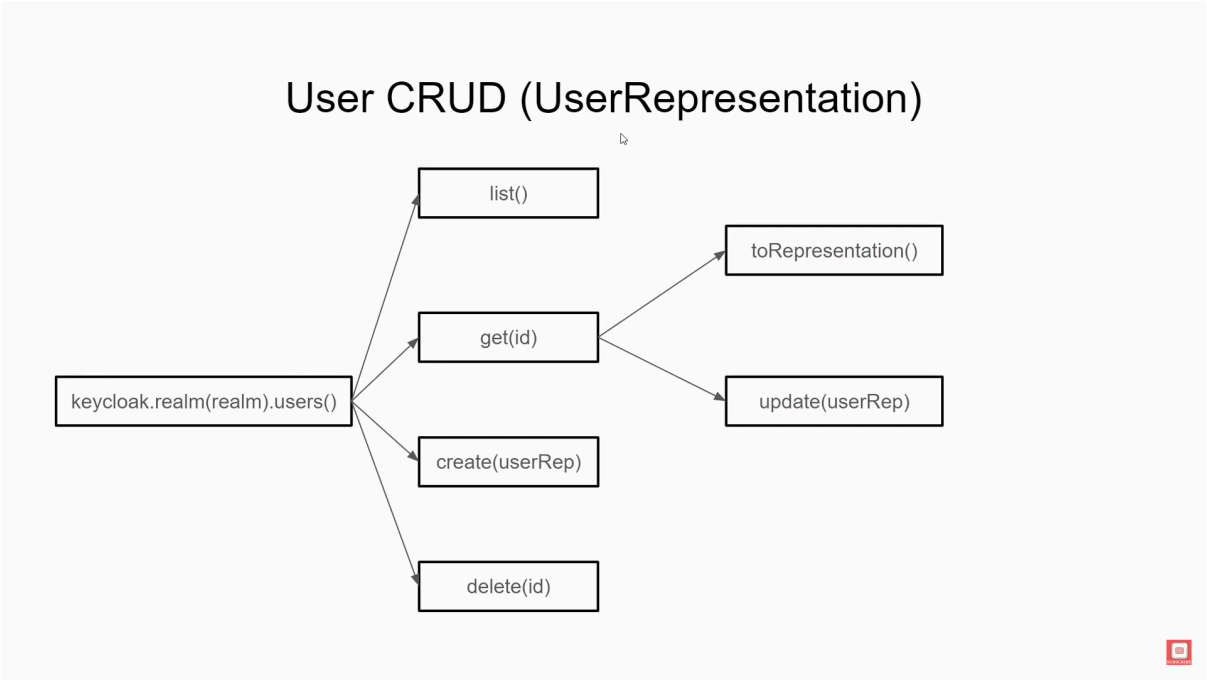

Activities
Firefox Web Browser
Feb 20 9:57 PM
Keycloak Doc's - Google
Keycloak Administration
type of models in spring
localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/permission/resource/4b36b8a2-3f42-42d0-99c5-f9209e9f
KEYCLOAK
admin
springboot-be
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Clients > Client details > Permission details
keycloak By Admin Permission
Action
Name *
keycloak By Admin Permission
Description
Apply to resource type
Off
Resources *
keycloak By Admin Policy
Policies
keycloak By Admin Po..
Decision strategy
Affirmative
Unanimous
Consensus
Save
Cancel
localhost:8081/admin/master/console/#/springboot-be/clients/dd4361c6-a9a8-450d-bc07-a820731a7810/authorization/policy/e4061e52-d7dd-484a-9e2e-1ec8816b00c8/role



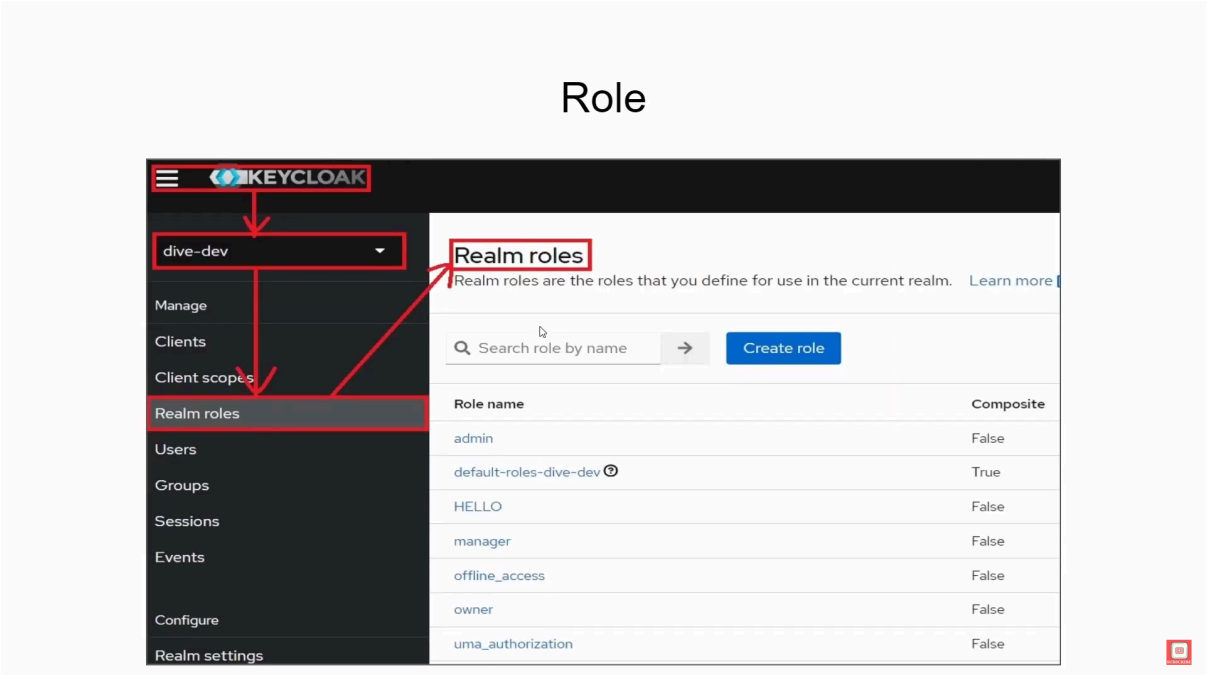
CRUD Operation Flow of Keycloak:

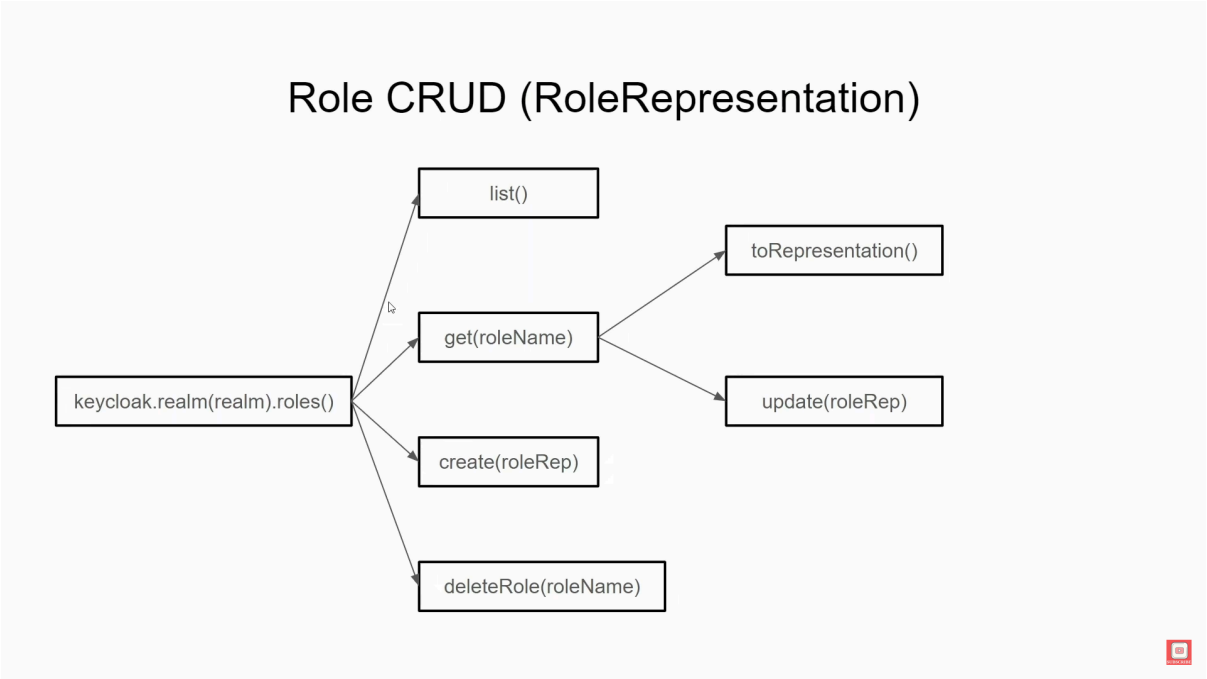
USER:



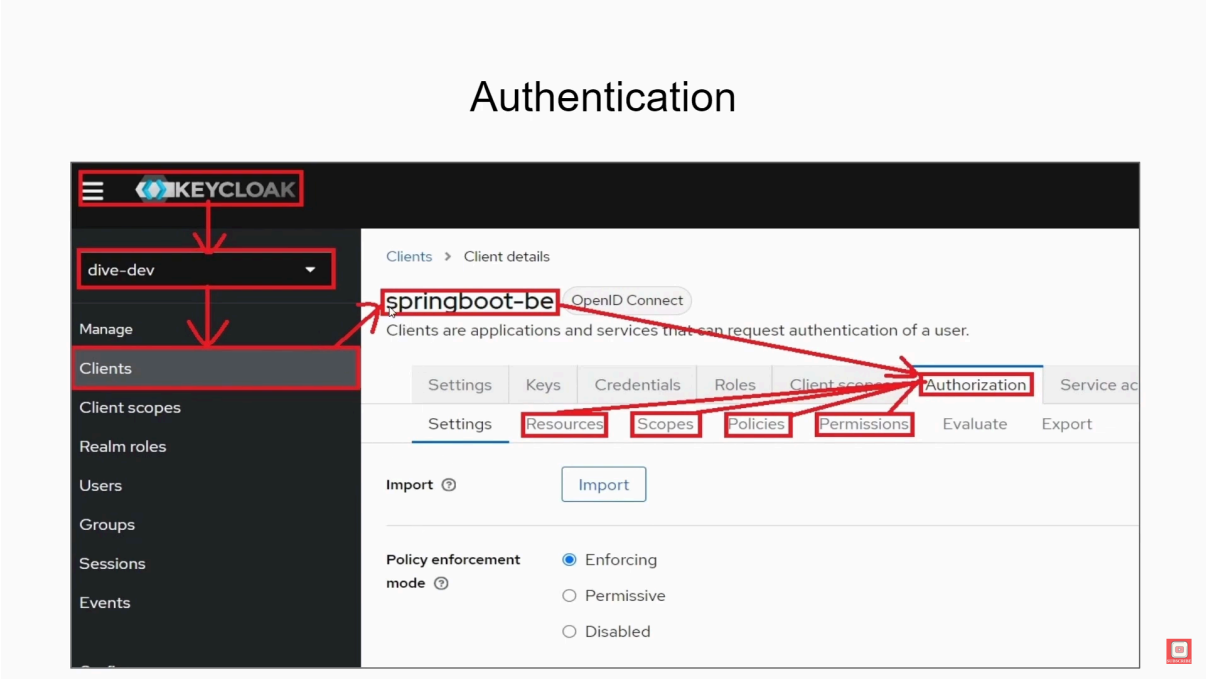

User CRUD (UserRepresentation)
keycloak.realm(realm).users()
list()
get(id)
create(userRep)
delete(id)
toRepresentation()
update(userRep)

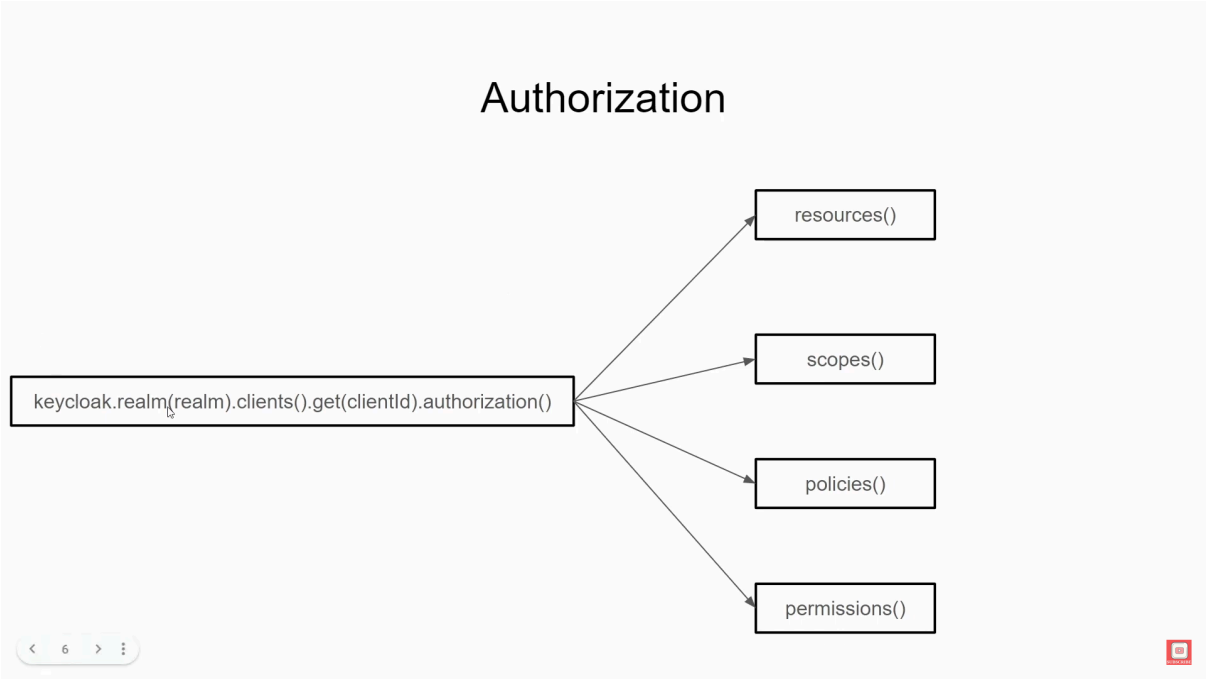
ROLES:


Role
KEYCLOAK
dive-dev
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Realm roles
Realm roles are the roles that you define for use in the current realm.
Learn more
Search role by name
Create role
Role name
Composite
admin
False
default-roles-dive-dev
True
HELLO
False
manager
False
offline_access
False
owner
False
uma_authorization
False

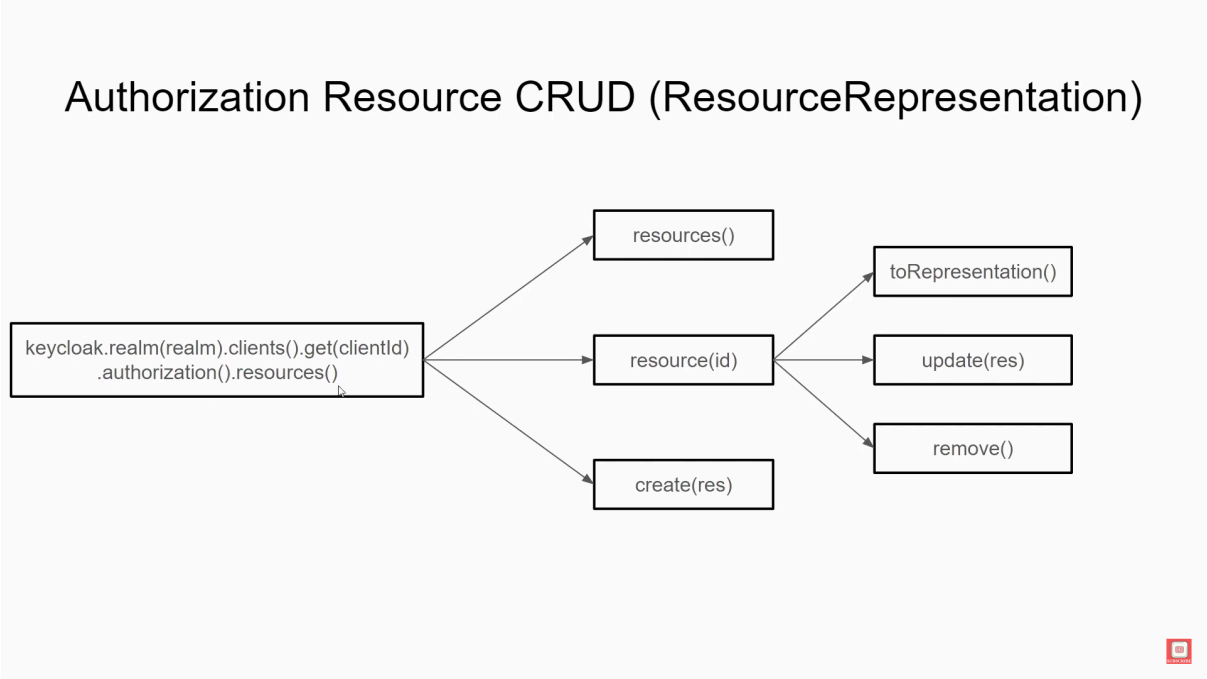


AUTHENTICATION:

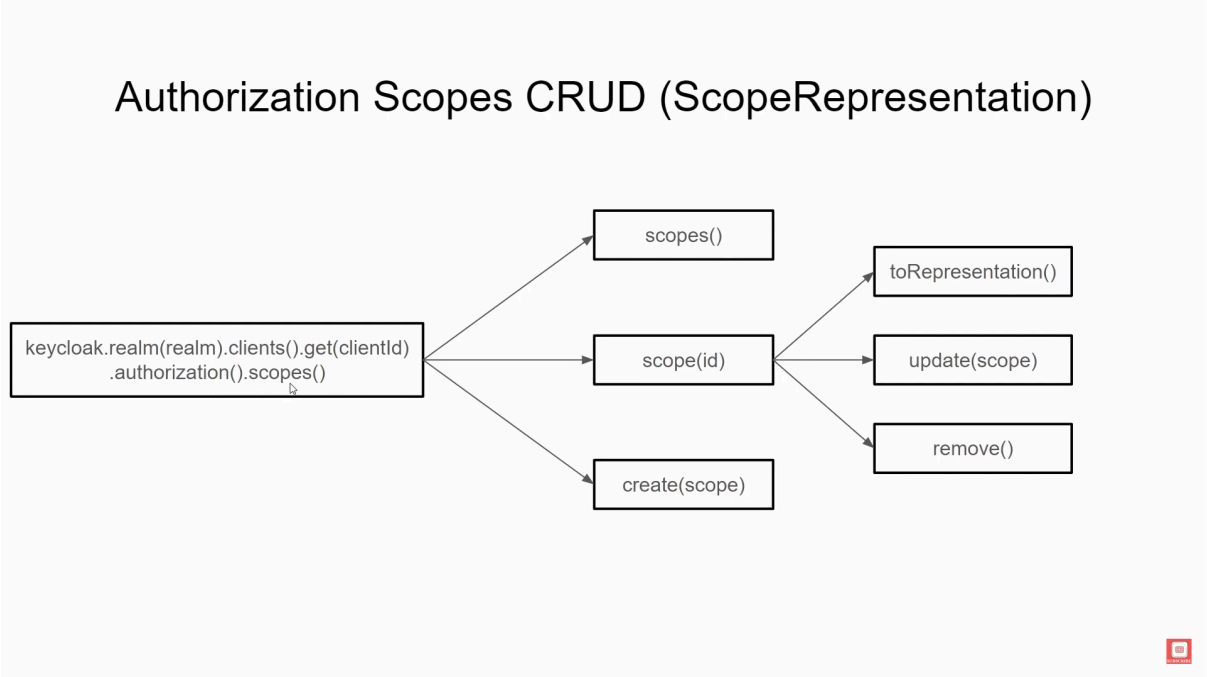



Authorization
keycloak.realm(realm).clients().get(clientId).authorization()
resources()
scopes()
policies()
permissions()
6

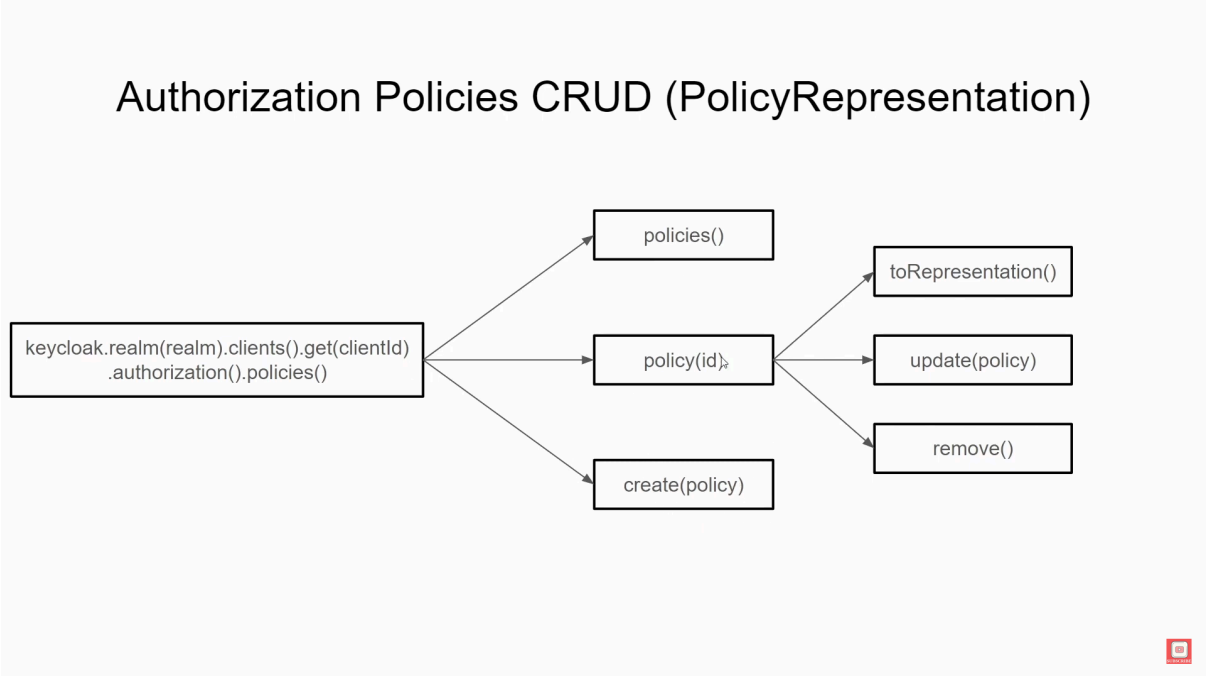
RESOURCES :


Authorization Resource CRUD (ResourceRepresentation)
keycloak.realm(realm).clients().get(clientId)
.authorization().resources()
resources()
resource(id)
create(res)
toRepresentation()
update(res)
remove()

SCOPE:

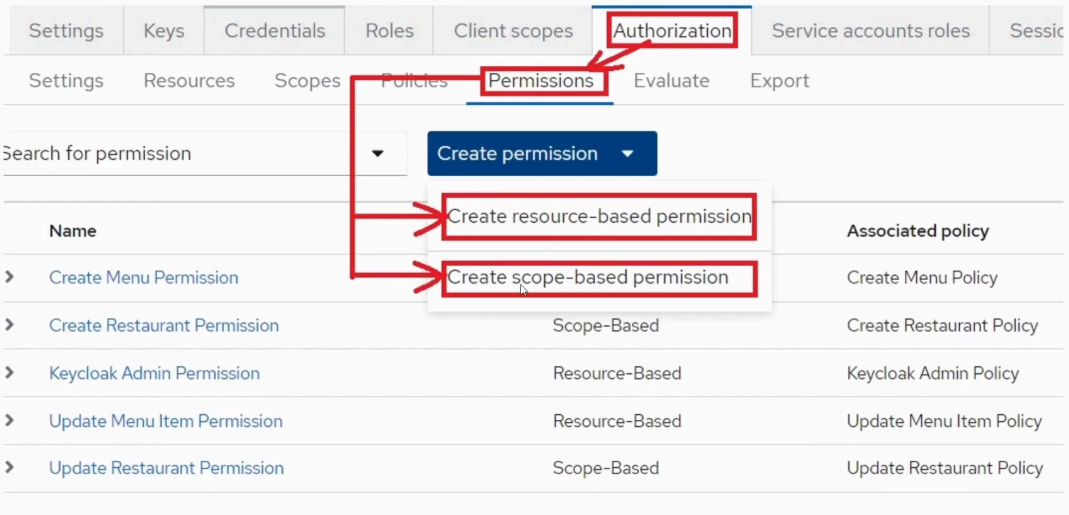


POLICIES:

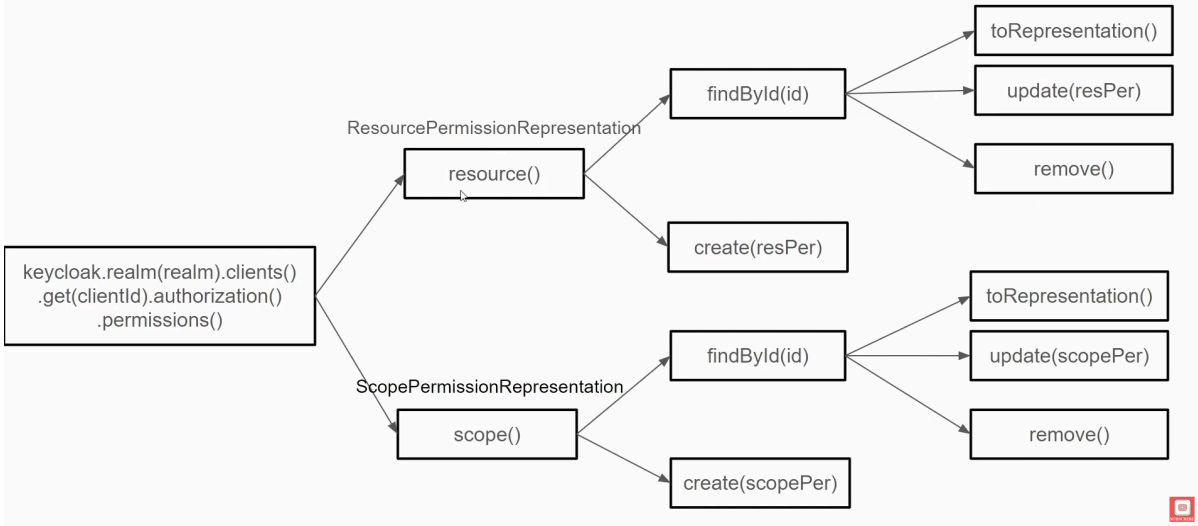


PERMISSION:

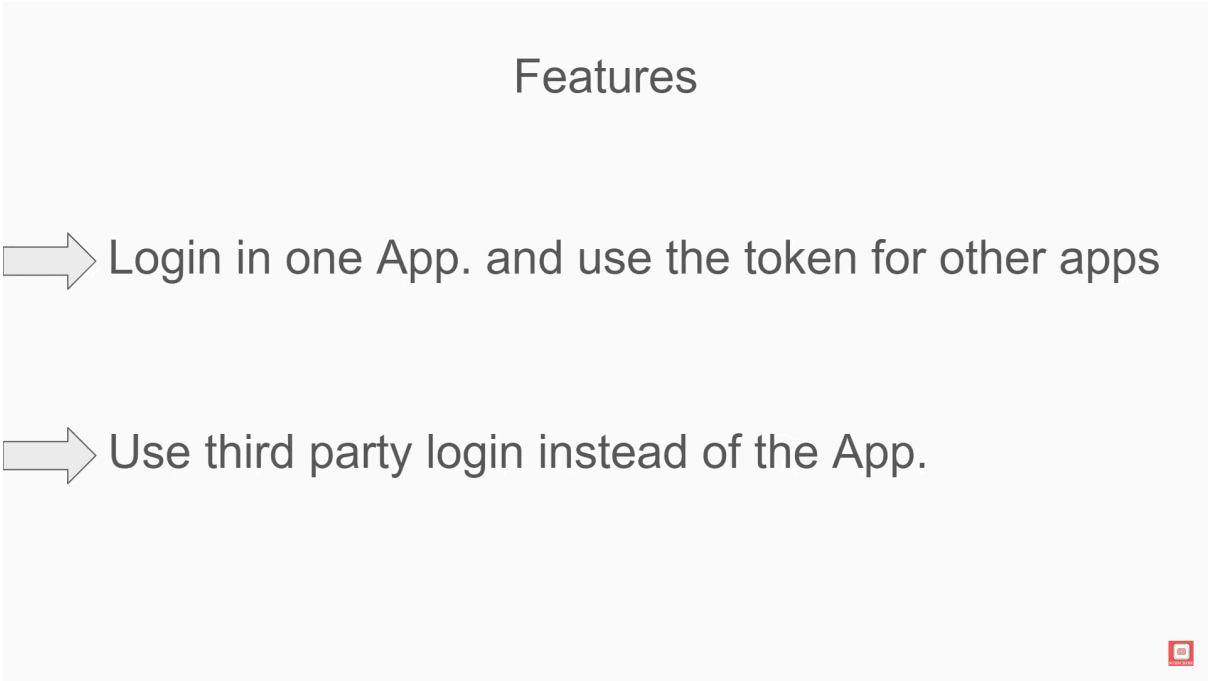
Authorization Permission



Authorization Permission CRUD



SSO (SINGLE SIGN ON):



Two types of sso is :

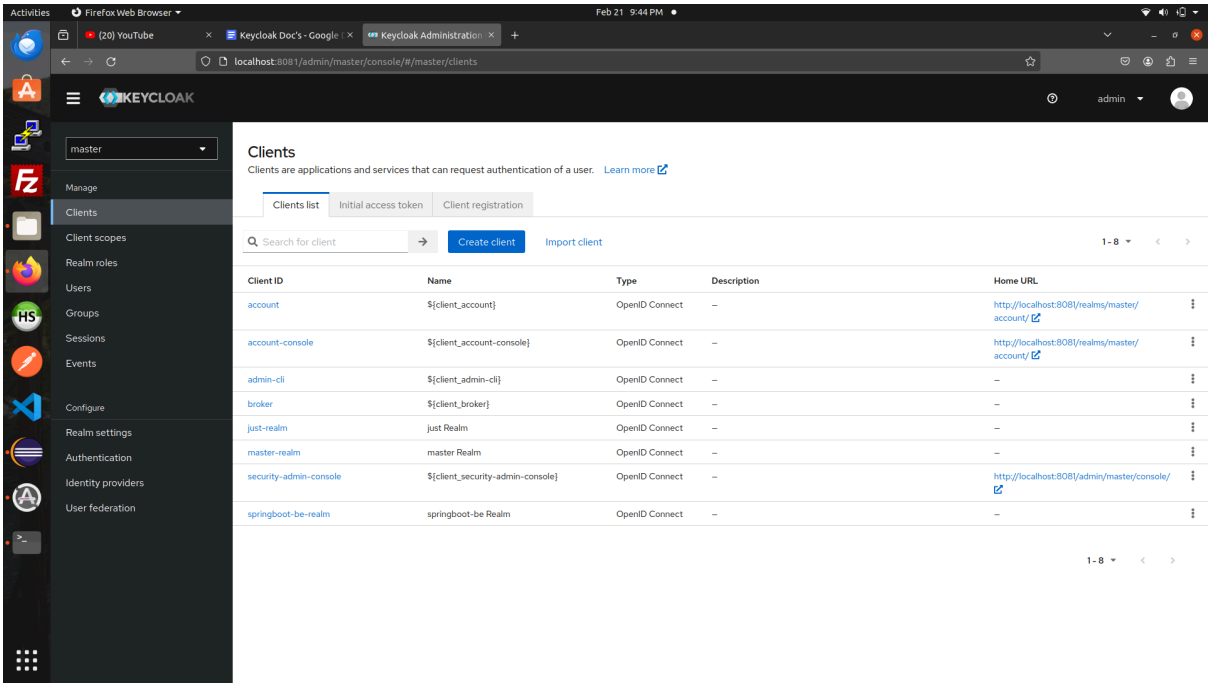
Login in one App Eg: If you have more than one product and Login for all users is the same,If you login to One of the products they can assess for other products.

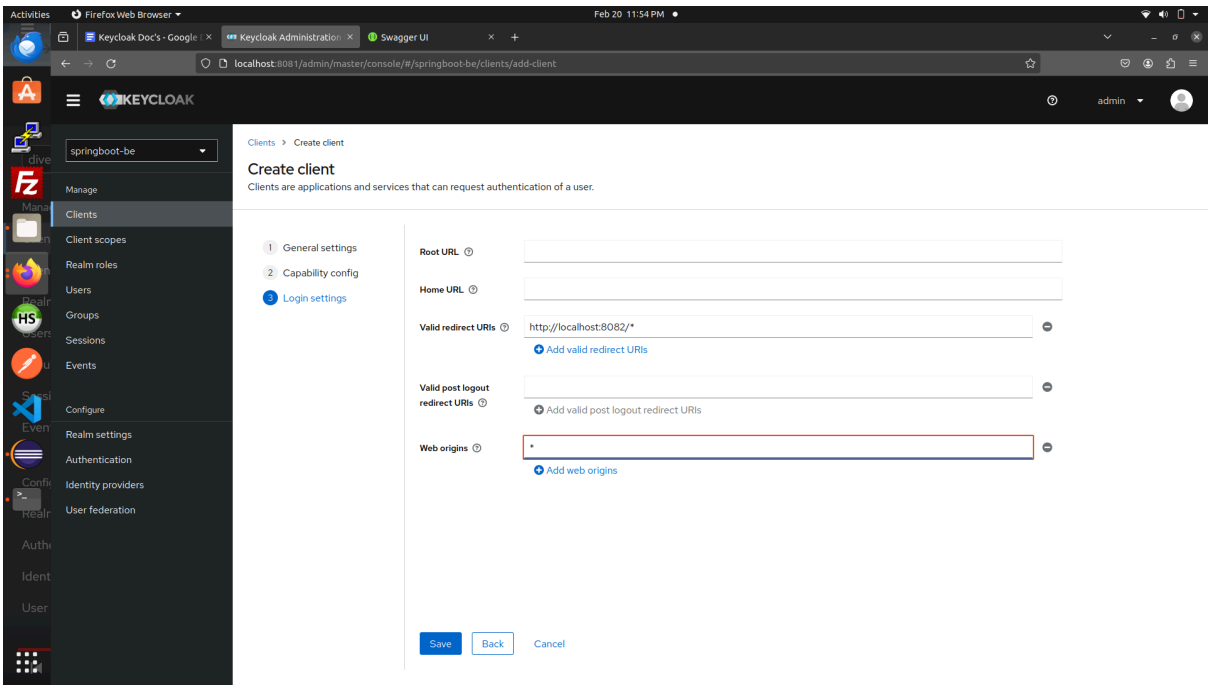
Use Third Party Login Eg : user can login by using Third party App (Github,Google,fb etc..)
(you can visit the YOUTUBE : Dive into development)

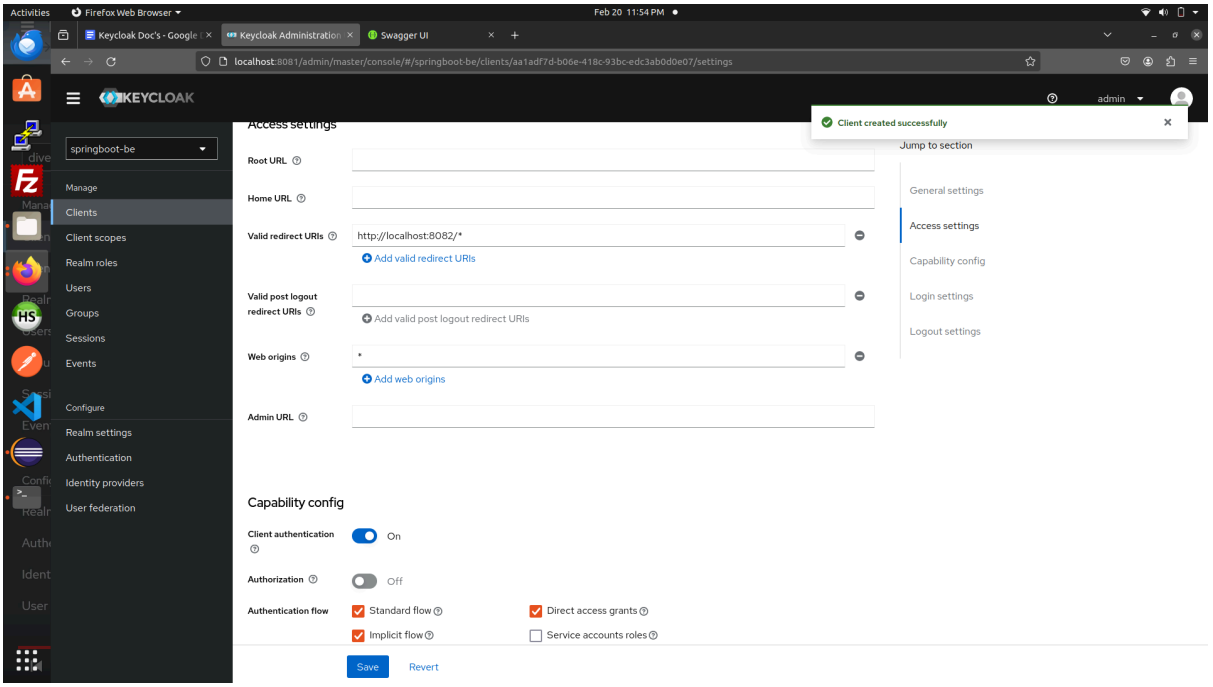
The Realm should be the same but Different clients need to be for sso.
EG: one origination(REALM) can be but different sectors(CLIENT).

Login in one App:

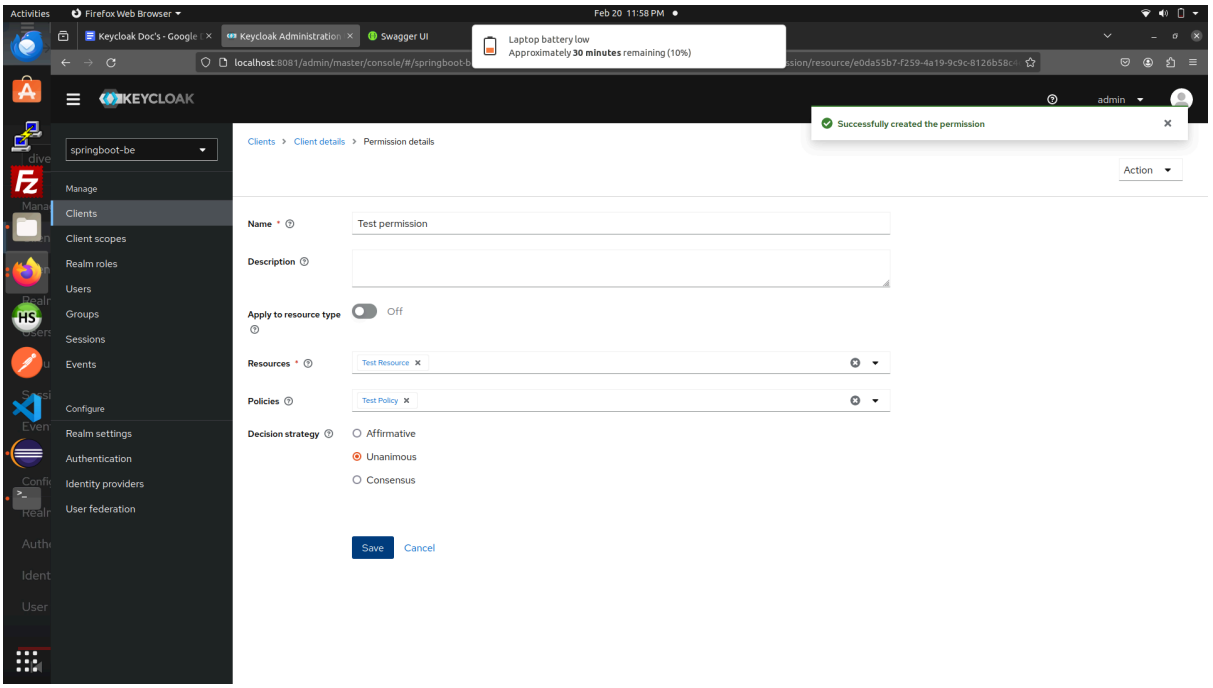
Create a client in the Realm.

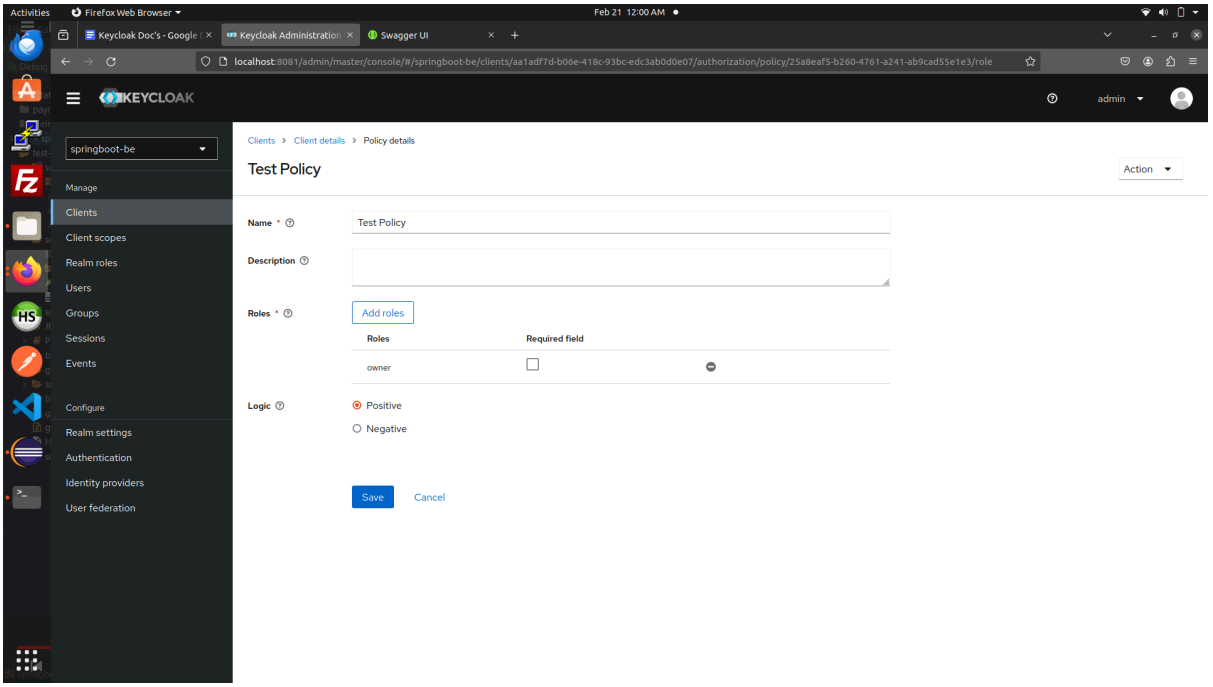


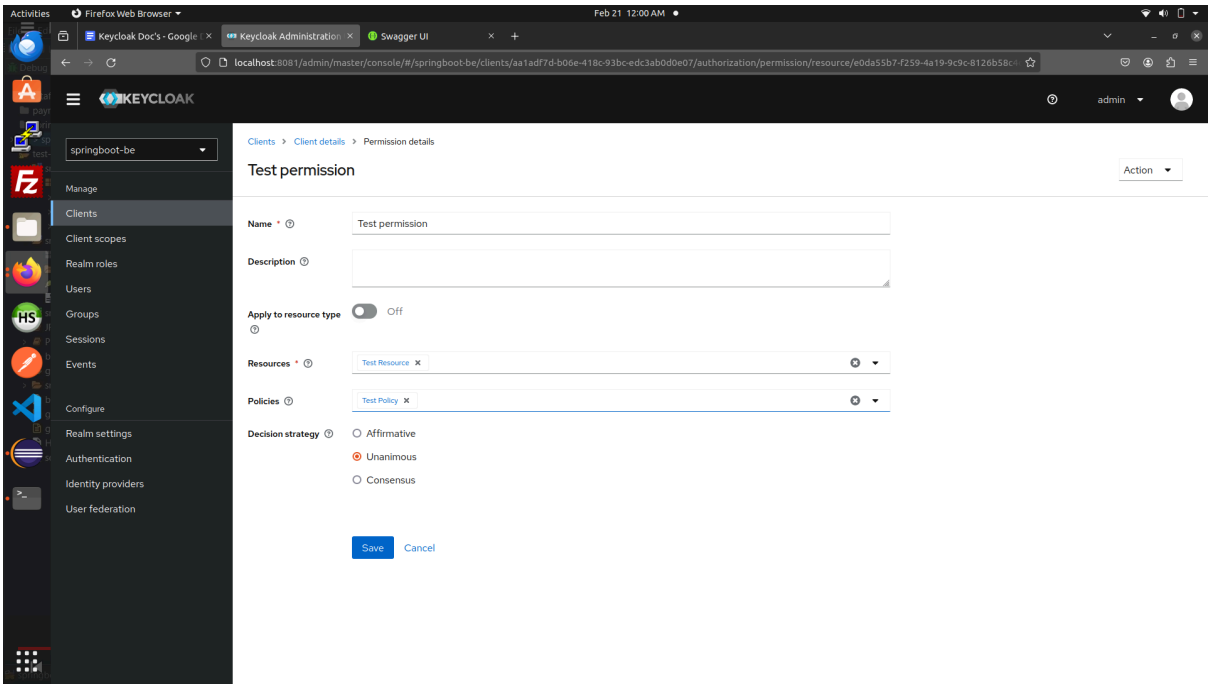




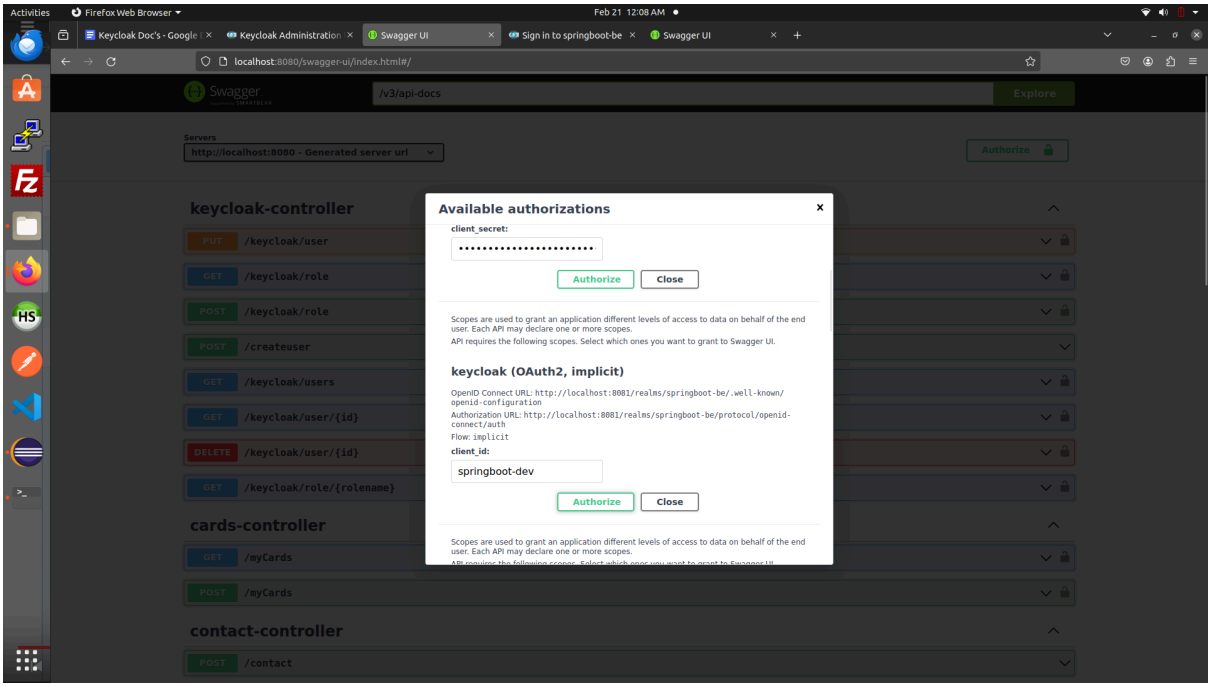
In Authentication:







By using **implicit Login we can achieve the SSO.**

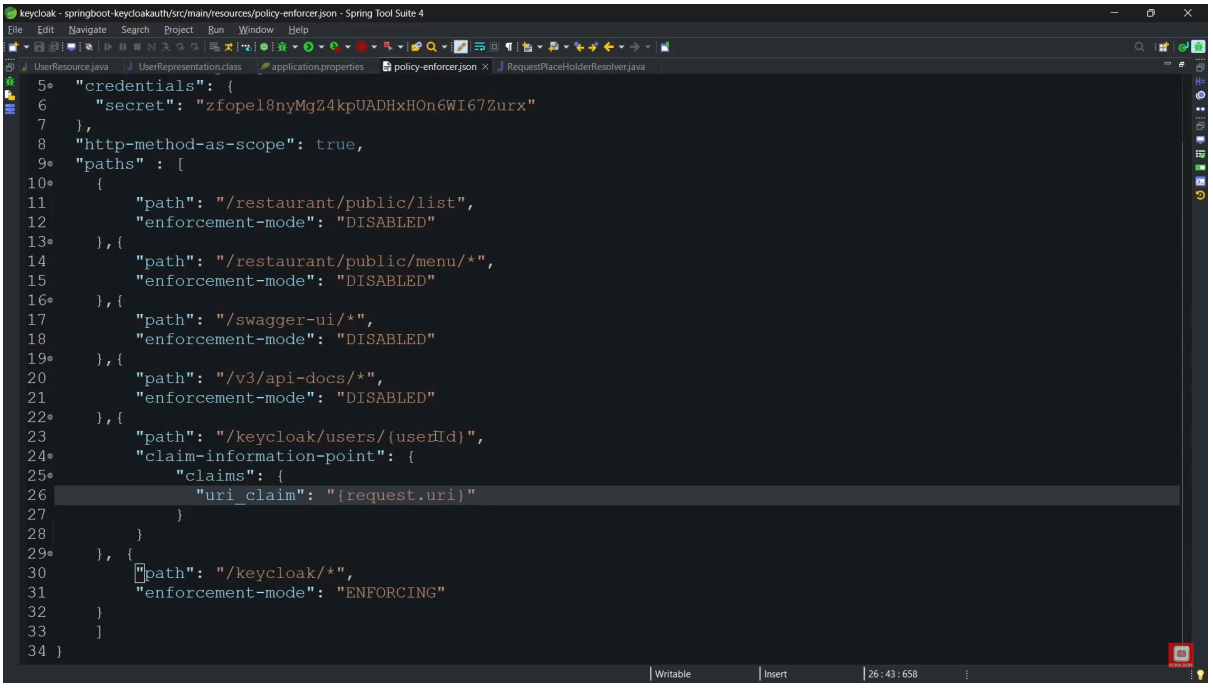


To Access ADMIN API for Other User :

(eg: if user login but in some cases users need to access the certain API's so this logic will be used).

This (/keycloak/user/{id}) API needs to be accessed by a certain user.

In Spring boot :



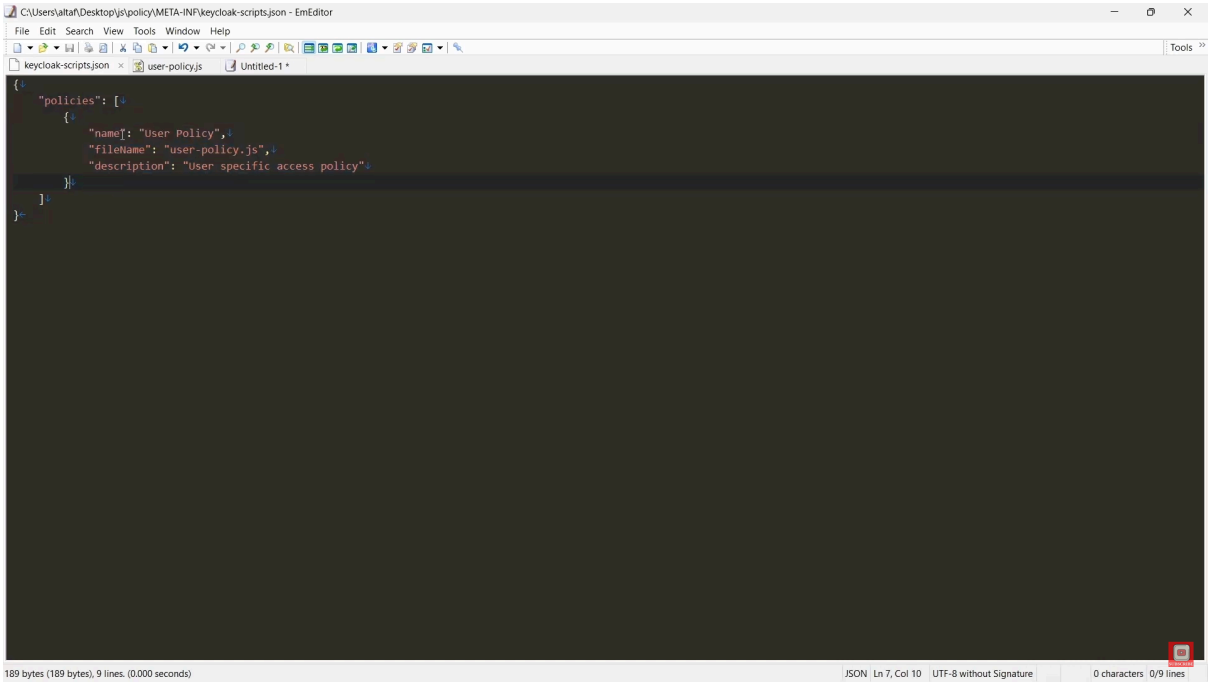
The /keycloak/user/{id} we enabling for other user to access and we need to set claim-info In the uri_claim we are setting the param variable,So we used {request.uri}.

If we add the /keycloak/user/{id} reset of the /keycloak API will not work so we will enforce the /keycloak/*.

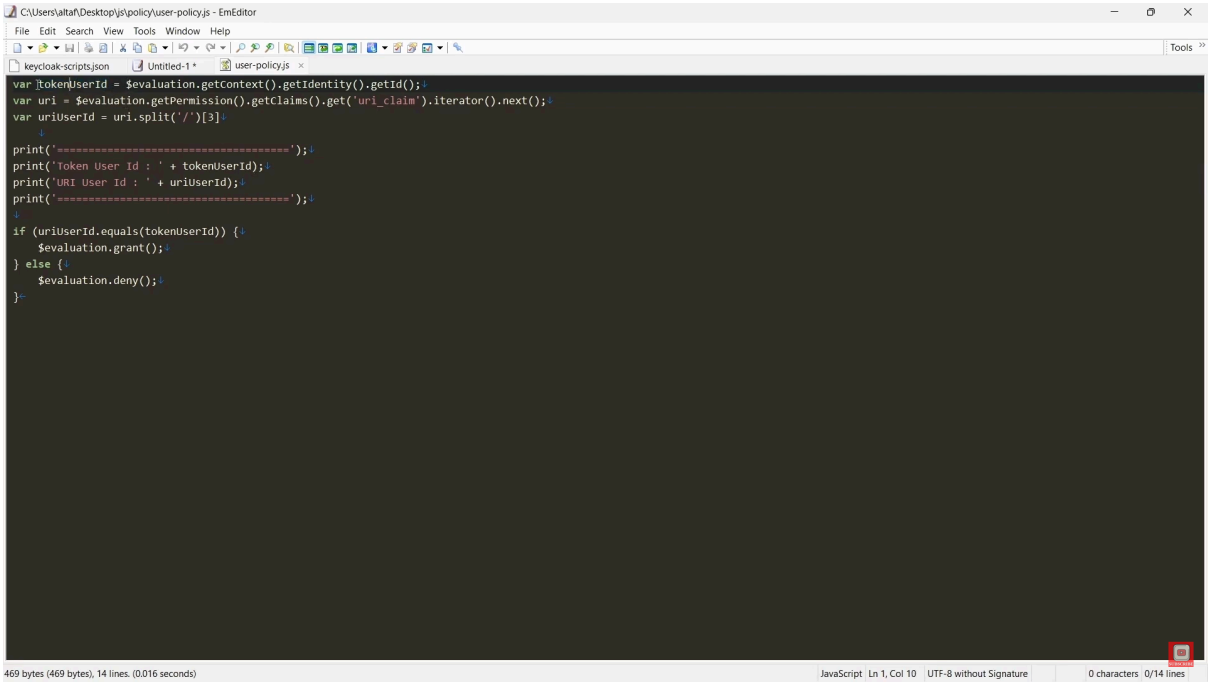
Need to create JavaScript Policy Provider:

Create a Folder to create the Files.

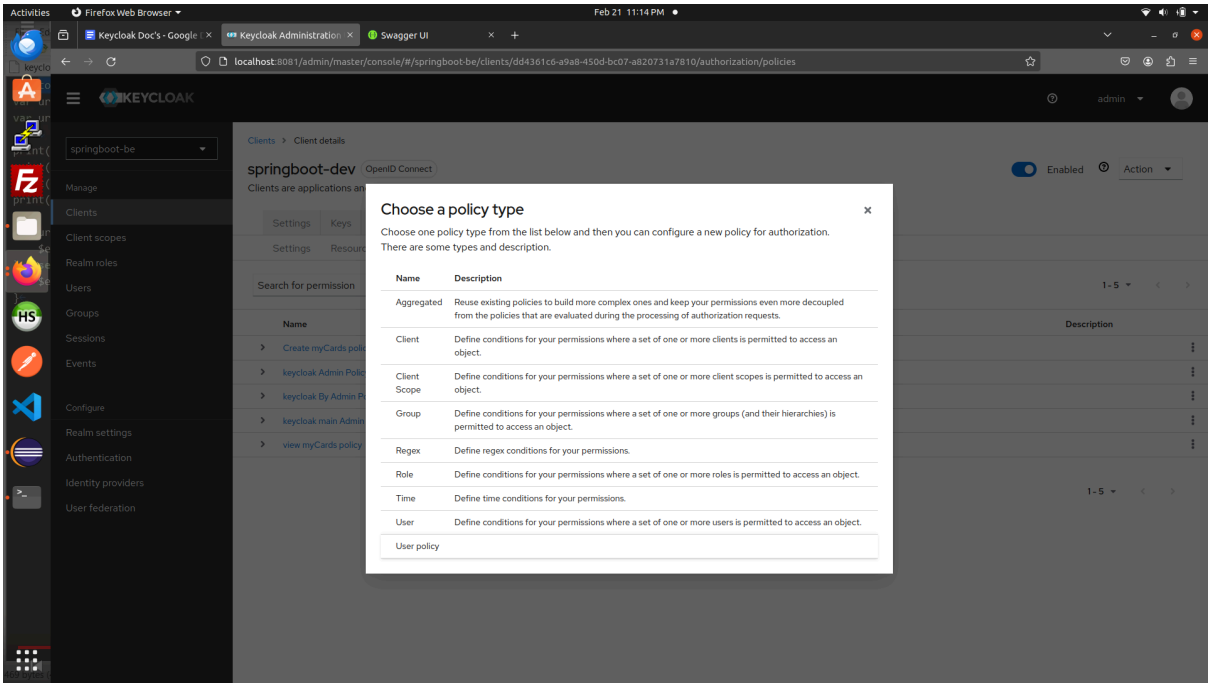
Create Folder name as META-INF , Inside the META-INF keycloak-script.json.
Open the keycloak-script.json file.



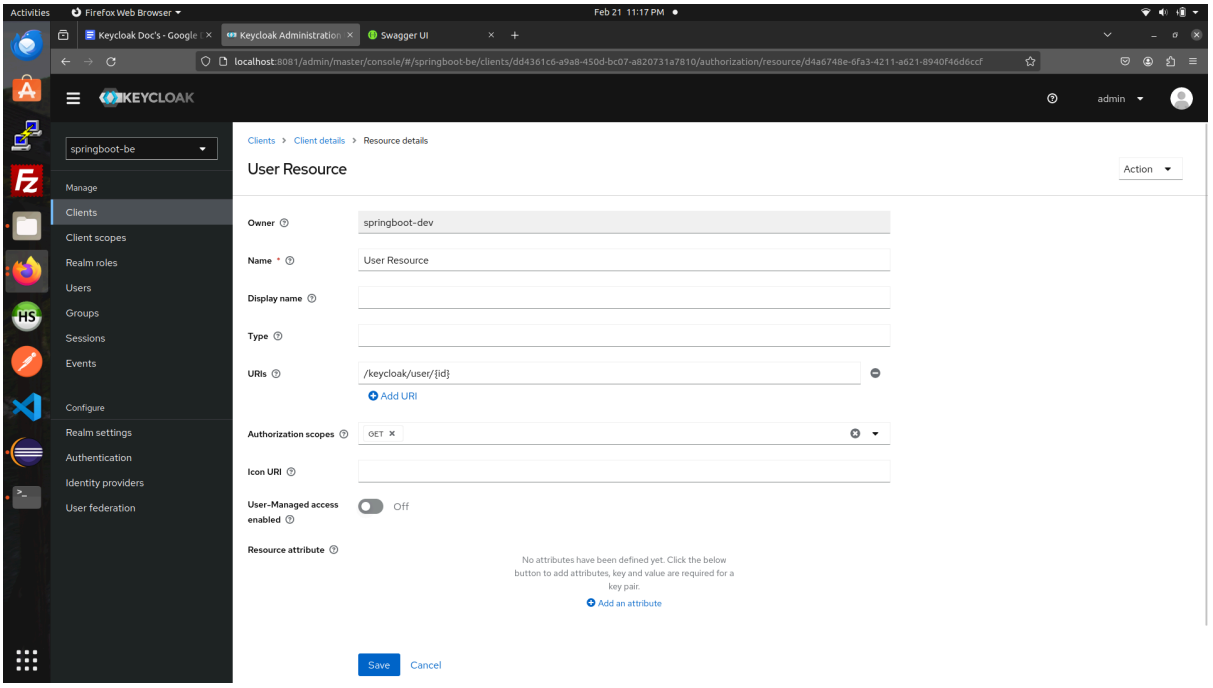
The filename is given as use-policy.js.
Go back to from META-INF and create user-policy.js.



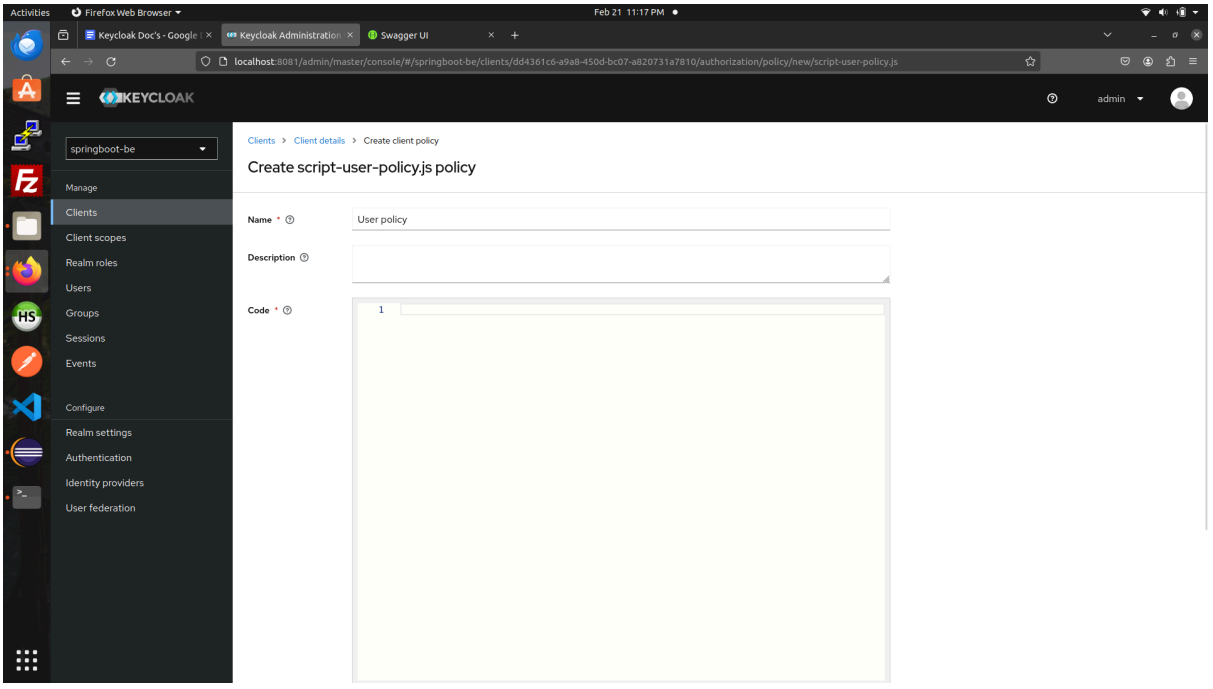
In the uri we are getting an uri-claim which we will mention in the Spring-boot json file.
Select the All folder and file compare to zip and change the exe to Zip to jar so jar file will be created.
Then copy the jar and place it in the /keycloak/providers.
Go -> claim -> Authentication-> Policies -> create policy you can see the User-policy.



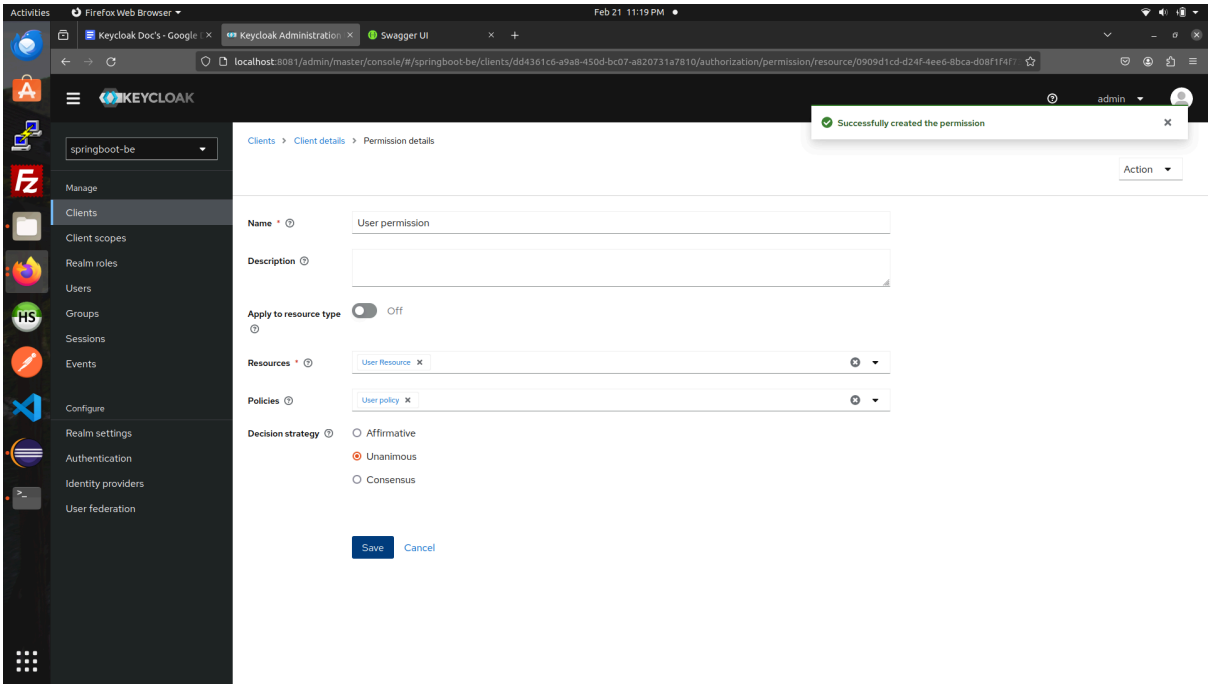
Create a Resource :



Create Policy (select the user-policy)

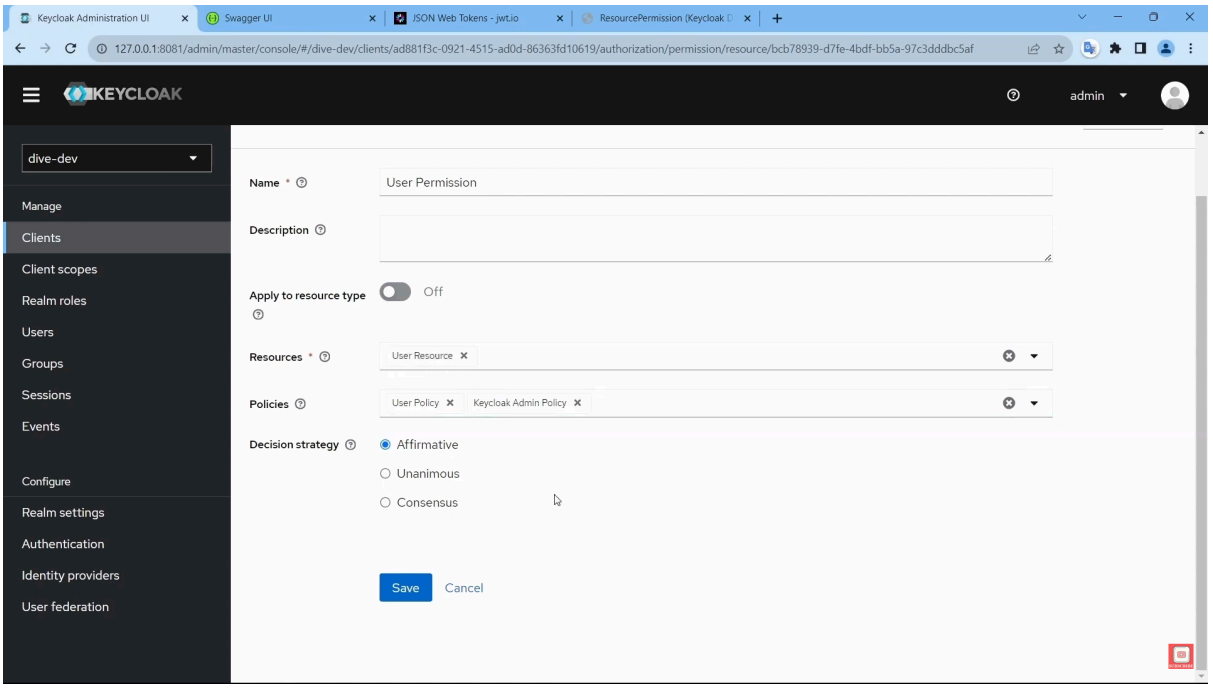


Create Permission:

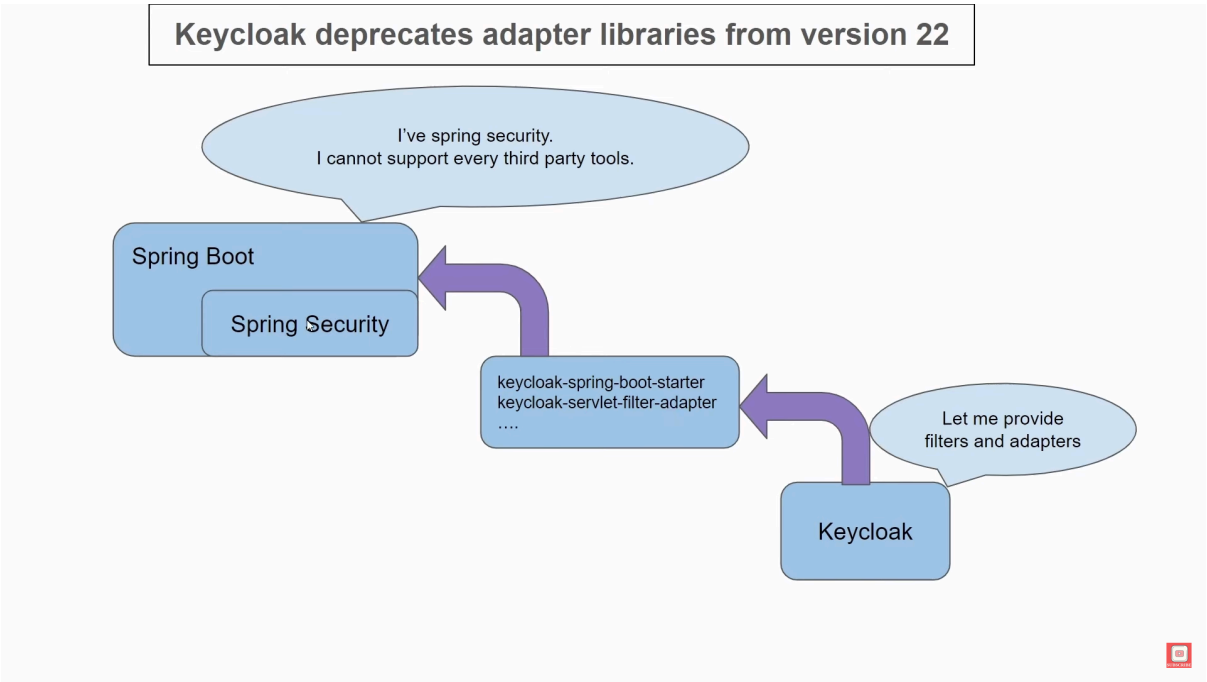


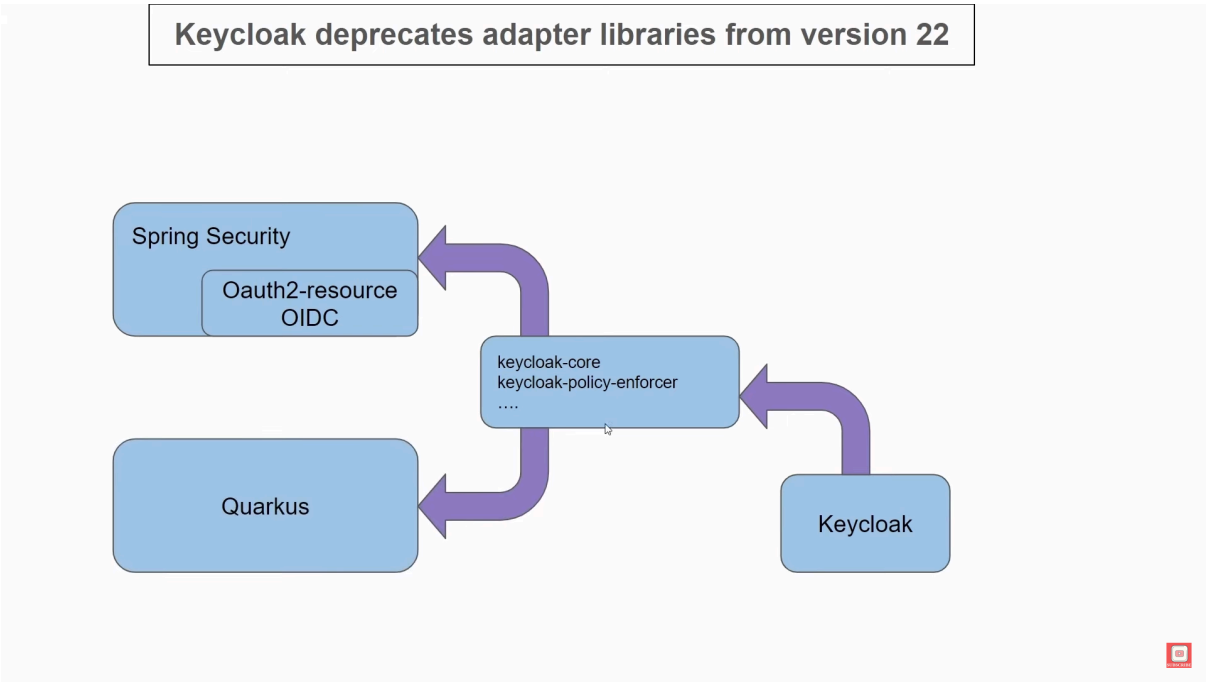
If we test the API's It will work for the same user Login and same UserID need to be equal then they can use the API.

If Admin needs access we can specify in the Permission.



If we add More than Policy we need to concentrate on Decision strategy.
Affirmative : Atleast one of Policy need to be satisfied.
Unanimous : All the Policy needs to be satisfied.





We can add Extra CLAIMS using spring boot:
View in github: